The London School of Economics and Political Science

From Components to Compositions: (De-)construction of computer-controlled behaviour with the Robot Operating System

Antti Kalervo Lyyra

A thesis submitted to the Department of Management, Information Systems and Innovation Group, of the London School of Economics for the degree of Doctor of Philosophy.

London, June 2018

Declaration

I certify that the thesis I have presented for examination for the MPhil/PhD degree of the London School of Economics and Political Science is solely my own work other than where I have clearly indicated that it is the work of others (in which case the extent of any work carried out jointly by me and any other person is clearly identified in it).

I declare that my thesis consists of 71549 words.

Abstract

Robots and autonomous systems play an increasingly important role in modern societies. This role is expected to increase as the computational methods and capabilities advance. Robots and autonomous systems produce goal-directed and context-dependent behaviour with an aim to loosen the coupling between the machines and their operators. These systems are a domain of complex digital innovation that intertwines the physical and digital worlds with computer-controlled behaviour as robots and autonomous systems render their behaviour from the interaction with the surrounding environment. Complex product and system innovation literature maintains that designers are expected to have detailed knowledge of different components and their interactions. To the contrary, digital innovation literature holds that end-product agnostic components can be generatively combined from heterogeneous sources utilising standardised interfaces. An in-depth case study into the Robot Operating System (ROS) was conducted to explore the conceptual tension between the specificity of designs and distributedness of knowledge and control in the context of complex digital innovation. The thematic analysis of documentary evidence, field notes and interviews produced three contributions. First, the case description presents how ROS has evolved over the past ten years to a global open-source community that is widely used in the development of robots and autonomous systems. Second, a model that conceptualises robots and autonomous as *contextually bound and embodied chains of transformation* is proposed to describe the structural and functional dynamics of complex digital innovation. Third, the *generative-integrative mode of development* is proposed to characterise the process of innovation that begins from a generative combination of components and subsequently proceeds to the integration phase during which the system behaviour is experimented, observed and adjusted. As the initial combination builds upon underspecification and constructive ambiguity, the generative combination is gradually crafted into a more dependable composition through the iterative removal of semantic incongruences.

Acknowledgements

I would like to express my profound gratitude to my supervisor Dr Carsten Sørensen for his continuous support, patience and motivation throughout this research. His knowledge and guidance have been of paramount importance throughout my PhD.

In addition to Carsten, I am in gratitude to the Information Systems and Innovation Group and its excellent faculty. Professor Jannis Kallinikos, Dr Will Venters and Dr Maha Shaikh have provided their valuable advice and support, and through various discussions and research seminars, I have learned to broaden my approach to research and to question many of my previous assumptions. I also thank my joyous community of PhD fellows, Kanchana Ambagahawita, Kari Koskinen, Emilio Lastra Gil, Kyung Ryul Park, Atta Addo, Florian Allwein, Gizdem Akdur, Zeynep Kaparoglu, Cody Dodd, Dana Lunberry, Rohit Nair, Andrea Paletti, Marta Stelmaszak, Erika Valderrama Venegas, Laura Zimmerman, Ceren Erdem, Rebecca Campbell, Boyi Li, Niccolò Tempini, Aleksi Aaltonen and Enrico Rossi, for stimulating discussions, friendship and support.

This PhD was funded by Liikesivistysrahasto, Emil Aaltosen Säätiö and LSE's Research Studentship Scheme, for which I am profoundly grateful.

Last but not least, there are no words to express the gratitude for the enduring support and kind encouragement I have received from my lovely wife Jane who always looks on the bright side. A few days after the viva, our son Leo Lyyra was born.

Table of contents

List of abbreviations

BSD – Berkeley Software Distribution
DARPA – Defence Advanced Research Project Agency
DDS – Data Distributions Services standard
DRC – DARPA Robotics Challenge
ERF – European Robotics Forum
ICRA – IEEE International Conference on Robotics and Automation
IREX – International Robot Exhibition
LCIM – Levels of Conceptual Interoperability Model
MRDS – Microsoft Robotics Developer Studio
NAO – A bipedal robot from Softbank Robotics
NASA – National Aeronautics and Space Administration
NRI – National Robotics Initiative
OMG – Object Management Group
Orocos – Open Robot Control Software
OSI – Open Systems Interconnection model
OSRF – Open Source Robotics Foundation
PR – Personal Robot (Stanford)
PR2 – Personal Robot 2 (Willow Garage)
ROS – Robot Operating System
ROSCon – ROS developer conference
ROS-I – ROS-Industrial consortia
Rqt – ROS QT user interface
RViz – Robot visualiser
STAIR – Stanford Artificial Intelligent Robot
TCP/IP – Transmission Control Protocol/Internet Protocol

List of tables

List of figures

Glossary of key terms

This glossary outlines central concepts and key terminology used in this thesis.

Robots and autonomous systems refer to machines and systems that produce goal-directed and context-dependent behaviour to operate autonomously with limited human intervention. They have sensors and actuators and they render their behaviour from the direct interaction with the surrounding environment. For the most part, their behaviour is controlled by computers.

Artificial intelligence refers to software. In particular to computational processes that transform between qualitatively different inputs to outputs in a way that can be viewed as advanced and sophisticated.

A robot system refers to some particular instance of robots and autonomous systems. A self-driving car or an industrial robot can be considered as a robot system.

Product architecture refers to a scheme by which the functionality of a product is allocated to its components. Can be characterised for example in terms of modularity and integrality. *With modularity*, functional interdependencies among components are low so that they can be separated and combined to produce product variety. With *integrality*, functional interdependencies among components are high so that a change in a component triggers changes in the components it interacts with.

Modularisation refers to a top-down process that partitions a product design into modules that can be separated and combined as per the design rules. Assumes centralised design agency from the part of the architect that performs the partitioning into a hierarchy of parts. Characterises coordinated product innovation.

Generativity refers to a bottom-up approach where new assemblages of digital products can be generatively created from end-produce agnostic

components and systems that adhere to common standards and open interfaces. Assumes distributed design agency as the knowledge and control of digital objects and artefacts is distributed across a heterogeneous group of actors. Characterises uncoordinated digital innovation.

Specificity of complex systems refers to a dual-view of product architectures that entails the hierarchies of inclusion and control. The hierarchy of inclusion refers to a hierarchy of parts that forms the physical embodiment, and the hierarchy of control refers to the parts and systems that control the behaviour of that embodiment. As the two hierarchies are interdependent, detailed knowledge of components and their interactions is needed to produce desired functionalities.

Organising logic of innovation refers to the dynamics of combination in the context of product architectures. Modularity (modularisation), generativity and specificity (complexity) manifest different logics of combination.

Complex digital innovation is a form of digital innovation that intertwines the physical and digital worlds with computer-controlled behaviour. The organising logic of complex digital innovation is the focal point of this research.

A system is a complex of interacting elements. Systems can be differentiated between each other by the number and type of their constituent elements and their respective interactions.

Simon's theory of hierarchy postulates the structure of complex systems as nested and recursive hierarchies that are nearly decomposable. The hierarchies consist of sets of subsystems that in combination form some higher-level system. A *subsystem* is a frame-dependent concept and what is considered as a sub or super system depends on the observer's frame of reference. In general, an element, component, part or subassembly can be considered as a subsystem. A *combination* is an instance of two or more subsystem interacting with each other to produce some higher-level system. In a way, a subsystem and combination are the two sides of the same coin.

The Robot Operating System (ROS) is an open-source robot software development framework. The ROS communication system, development tools and software distribution infrastructure facilitate the development of robots and autonomous systems as distributed computation, and the ROS community brings together a heterogeneous group of roboticists from academia and industry.

Thematic analysis is an iterative and interrogative method of data analysis that seeks to identify recurring patterns that warrant categorisation, abstraction and conceptualisation of phenomena. This method of analysis consists of five phases, which are familiarisation, open coding, categorisation, thematisation and conceptualisation.

Proposed concepts

Contextually bound and embodied chains of transformation is a notion that conceptualises the structural-functional characteristics of robots and autonomous systems that explain the organising logic of innovation in the context of complex digital innovation. This seeks to highlight that the appropriateness of behaviour is context-dependent, the behaviour and computation are embodied, and that computation is composed of the processes of communication and transformation. All these factors play part in the dynamics of combination.

Generative-integrative mode of development is an approach to systems development. It characterises the process of complex digital innovation that begins from a generative combination of components and subsequently proceeds to the phase of integration during which the system behaviour is experimented, observed and adjusted. As the initial combination builds upon underspecification and constructive ambiguity, the generative combination is crafted gradually into a more dependable composition through the iterative removal of semantic incongruences.

1 Introduction

The field of robots and autonomous systems is a prominent but understudied domain of digital innovation. Much of the innovation in the field derives from the rapid progress of the digital computing technologies, such as pervasive connectivity and increasingly sophisticated computing methods and techniques, which are often referred to as artificial intelligence and machine learning. These advances enable the design and production of robots and autonomous systems that can be deployed to carry out tasks on their own with limited human intervention. Although digital innovation researchers have examined a range of matters, such as digital infrastructures (Tilson et al. 2010), digital platforms in their different forms (de Reuver et al. 2017), digitised and connected products (Yoo et al. 2010; Henfridsson et al. 2014), social media (Alaimo & Kallinikos 2017), mobility and digital service (Barrett et al. 2015), the innovation in the field of robots and autonomous (robotics) has attracted little attention to date.

To begin to bridge this gap, this research sets out to study *the organising logic of digital innovation in the context of robots and autonomous systems* to explore to what the extent the current conceptualisations of organising logics apply in the field of robotics. Considering that different domains and phenomena of digital innovation are related and complementary to each other, the introduction begins by placing this research against the broader backdrop of digital innovation. Then, the notion of organising logic is discussed before outlining the motivation, research question, empirical context and objectives of this research.

The emergence of digital information and communication infrastructures have transformed economic, organisational and social arrangements (Varian 2010), and digital technologies are expected to play a significant role in the future as well. A report by McKinsey Global Institute (Manyika et al. 2013) outlines twelve disruptive technologies of the future and provides a forecast of their economic impact. Out of the twelve technologies listed in the report, the six top places were occupied by digital or digitally driven technologies: mobile internet, automation of knowledge work, the Internet of Things, cloud computing, advanced robotics and autonomous and near-autonomous vehicles. The report

forecasts their total economic impact to be somewhere between 15.2 and 36.3 trillion by 2025. Although future-looking reports are prone to be wrong in details, timelines and sometimes in fundamentals, it would be surprising if the advances in digital technologies and automation would not play a significant part in/on future economic arrangements.

Technologies, in general, are created to serve some human purpose (Arthur 2009), and digital technologies are no different in this sense. Since the introduction of the digital computer in the 1950s, Information and Communication Technologies (ICTs) have been studied, designed, developed and deployed to augment human capabilities and to replace human labour in different work organisations (Avgerou 2000). Information and communication technologies are deeply embedded in modern organisations, and it would be difficult to imagine them without bringing digital technologies into service. However, digital technologies can be used to serve a variety of purposes which go beyond the traditional management and coordination of organisations' workflows, assets and labour through the processes of transmitting, storing and processing transactional data and information resources.

Digital and data-driven business organisations seek to transform the underlying business models (Bharadwaj et al. 2013) by developing novel ways for delivering goods and services in digitised forms and over digital networks (Tilson et al. 2010). While doing so, they also often seek to gather large data sets from diverse sources and analyse them using computational methods and tools in the hope of obtaining competitive advantage. The analysis of large data sets may provide useful insights for decision-makers, and certain knowledge-based tasks and jobs tasks that entail well-definable process of analysis and decision-making can be fully delegated to algorithms. This way, the use of computational methods can be used to support and automate knowledge work in organisations.

The impact of digitalisation extends also to products (Henfridsson et al. 2014), cyber-physical environments (Broy et al. 2012). They are becoming more digitised and connected to the communication networks and digital service infrastructures. For example, much of the functionality of modern cars is produced and controlled through digital means (Lyyra & Koskinen 2016), either

by the computer and software built into the car itself or by the cloud-based services that render services remotely (Svahn et al. 2017). In addition to products which can be considered as having clearly observable boundaries, such as cars, the Internet of Things combines the physical and digital aspects of products and services while blending into the physical infrastructures and environments in a ubiquitous manner (Conti et al. 2012). Broadly speaking, the Internet of Things can be described to consist of physical devices and products that can be connected to the Internet, such as thermometers, fridges and cars among many others, and of the communication networks and digital infrastructures, which facilitate and coordinate the exchange of data between different devices, products and cloud-based services (Broy et al. 2012).

The domain of robots and autonomous systems differs from other domains of digital innovation by its emphasis on the development of machines that perform goal-directed tasks in relation to the surrounding environment with limited human intervention (Siciliano & Khatib 2008; Bekey 2005). In order to loosen the coupling between the machines and their human operators, machines need to be equipped with mechanisms that are capable of producing and controlling machine behaviour in a way that is congruent with their tasks and task environments (Bissell 2009). To this end, researchers and developers of robots and autonomous systems integrate a range of digital technologies and methods and develop them further with the aim of developing computational models of behaviours that are robust and reliable (Bonsignorio & del Pobil 2015). This behavioural aspect of computation renders this domain of digital innovation extremely heterogeneous (Russell & Norvig 2010). The types and uses of robots and autonomous systems range from advanced manufacturing in production facilities and self-driving cars in public roads to Mars rovers and space exploration, to provide a few examples. The heavy reliance on digital technologies brings robots and autonomous systems into the purview of digital innovation (Nambisan et al. 2017).

Technological progress expands gradually the scope of tasks and jobs that can be transferred from humans to machines. As has happened throughout the history of technological and economic development (Arthur 2009), this expansion has kept the multifaceted set of boundaries of work, revenue and

control among machines, workers, developers and owners in the flux for the foreseeable future.

To succeed in competitive markets, firms and organisations from different industries are working towards increasing levels of autonomy by amalgamating digital technologies, methods and services with physical products (Nambisan et al. 2017; Fichman et al. 2014), and they appear to approach the amalgamation from different starting points and focus on different industries and market segments. Technology companies with a background on Internet and digital business models are inclined to integrate general and scalable computational methods into consumer-facing products, whereas organisations with a background on products, manufacturing and industrial services tend to focus on improving their products and service portfolios by making a better use of digital technologies and automation. To exemplify, Internet and consumer-oriented technology companies are racing to introduce voice-controlled user interfaces and digital assistants. They are being made available by Google (Google Assistant), Amazon (Alexa), Apple (Siri), Microsoft (Cortana) and Baidu (Duer), whereas the IBM Watson unit focuses on natural language processing services and automation of knowledge work in organisational settings. Google, an Alphabet business unit which focuses on search and advertising, announced a strategic focus from the mobile devices first to the artificial intelligence first to emphasise its efforts to leverage sophisticated computing methods, while another Alphabet business unit, DeepMind, focuses on the development and application of more general machine-learning and artificial intelligence methods and techniques which are capable of handling a wider variety of tasks and contexts. In the domain of products and services, self-driving cars have recently gained much momentum and attention. Traditional car makers, such as Volvo, Nissan and Audi, and their long-standing components suppliers compete with new entrants, such as Tesla, Über, Alphabet's Waymo, Mobileye, Oxbotica and Baidu, to develop technologies that could generate revenue from the automated act of driving (Thrun et al. 2006). For farming, companies such as Deere & Company and Bosch's Deepfield Robotics seek to automate fieldwork, whereas companies known for industrial automation, robots and services, such as Siemens, Atlas Copco, Metso Automation, ABB and Kuka among many others, are equipping their industrial and factory solutions with advanced

sensors, software and cloud-based technologies, and thereby expanding their product and service portfolios to increasingly autonomous cyber-physical systems and smart environments.

This cursory illustration shows how the boundaries between the digital, physical and autonomous technologies are becoming increasingly fuzzy and intertwined. Moreover, the number of firms and organisations investing in these technologies, in the hope of generating revenues from the work carried out by robots and autonomous systems, indicates the relevance of two underlying streams of this research. The first one of them is the amalgamation of the digital and the physical, and the second is the effort to develop products and services that exhibit increasing capabilities of goal-directed, task and context-dependent behaviour.

1.1 Motivation and scope of research

To date, the development of robots and autonomous systems has remained to a large part in the realm of well-resourced corporate and government entities that focus on specific products. However, the emergence of open-source communities that focus on robotics and artificial intelligence is challenging this by making more state-of-the-art software freely available. The ability to reuse, combine and build upon others' work lowers the bar of participation and opens up new possibilities, thereby allowing a broader spectrum of innovative and entrepreneurial firms and organisations, large and small, to innovate on robots and autonomous systems. However, while all this opens up appealing opportunities, it also simultaneously raises a question on how to coordinate and organise these novel and multifaceted avenues of innovation.

Organising logics of innovation can be conceptualised in various ways. In the context of this work, following Yoo et al. (2010), the notion of organising logic refers to the dynamics of combination in the context of product architectures. To present the motivation and scope of this research, the subsequent sections introduce the concept product architecture, describe it in terms of modularity and modularisation and then show how the applicability of modularisation has been questioned in two streams of innovation literature, of which one focuses on digital innovation and the other on complex products and systems.

1.1.1 Product architectures and organising logic of innovation

Products and systems are typically composed of different components, and product architecture defines the arrangement, functionality and interactions of components which in combination produce the overall functioning of a product (Ulrich 1995). Although product architectures may differ greatly from one to another, they can be characterised based on certain commonalities. Perhaps one of the most widely used characterisation of product architectures is presented along the spectrum between modularity and integrality (Ulrich 1995). This indicates the level of synergistic specificity among components and the extent to which they can be separated and recombined to produce a variety of products (Salvador 2007; Schilling 2000). A desired degree of modularity can be pursued through the centralised top-down process of modularisation (Baldwin & Clark 2000), which allocates the functionality of a product to its constituent components and specifies the design rules which govern the interconnections and interactions among components. Moreover, modular structures can be described to consist of platforms and platform complements (Baldwin & Woodard 2008); the platform consists of central and relatively stable components, whereas the complements that adhere to design rules and attach to the central components through well-specified interfaces exhibit a greater degree of variety (Salvador 2007). This way, modularisation and the specification of design rules provide the methods of partition and coordination that facilitate the distribution of design and manufacturing activities across organisational units and industrial ecosystems (Sanchez & Mahoney 1996). Subsequently, modular product architectures have come to bear significant implications in the division and organisation of innovative and productive labour in different organisations and industries (Baldwin & Clark 2000).

1.1.2 Distributedness of knowledge and control

Digital innovation literature questions the extent to which the notions of modularisation and modular product architectures apply to digital innovation. The main discontent is that the process of modularisation assumes a centralised design agency (Nambisan et al. 2017) that partitions the overall design of a product to components that can be designed and manufactured in a distributed manner before final assembly (Yoo 2012b). Therefore, the literature on digital

innovation approaches the process of design from a different direction and argues that digitised products emerge partly as generative combinations (Yoo et al. 2010) in the absence of centralised design agency (Nambisan et al. 2017). This is theorised to follow from the commodifying and generative characteristics of operating systems and communication protocols (Zittrain 2008; Zittrain 2006), which make it possible for broader audiences to take part in digital innovation. In this view, digitised products are seen as layeredly-modular (Yoo et al. 2010). This notion combines the concepts of products architectures as modular arrangements and as layered stacks. In the computer and software industry, system architectures are commonly presented as layered and hierarchical stacks. The stacks are collections of computer programs (components), where each of the components carries out a specific set of functions and in combination they produce the overall functioning of a computer. The combination of components that carry out different functions and reside at different levels of hierarchy relies on well-specified interfaces, standards and protocols (Yoo et al. 2010) which facilitate the communication between the components. This allows the separation of concerns by hiding of the internal functioning and complexity of components (Parnas 1972), which in turn facilities the divisions of innovative work as software designers can focus on their particular area of work. An operating system provides a prime example of hiding complexity; it hides the intricate details and complexity of computing hardware while making it accessible to software developers and applications (Tanenbaum & Bos 2014). This way, as long as different components adhere to specifications and produce their intended functionality, it is possible to combine ensembles of software components even if they originate from different sources (Yoo et al. 2010). While the reliance on pre-specified interfaces and layers of abstraction facilitates the division of labour and combination of components that originate from different sources, it also leads to a situation where knowledge and control of digital components and their internal functioning is distributed over a number actors and organisations (Yoo et al. 2010) while remaining open-ended and subject to frequent modifications (Kallinikos, Aaltonen, et al. 2013; Garud et al. 2008). Therefore, the digitalisation of products (Yoo et al. 2010), platforms (de Reuver et al. 2017) and infrastructures (Tilson et al. 2010) not only opens up new ways to divide and participate in the

innovative work, but it also challenges the established theories and practices of organisation and coordination.

1.1.3 Specificity of designs

Whereas the emergent and generative characteristics of digital innovation challenge the organising logic that builds on the notions of centralised design agency and modularisation, the literature on complex products and systems also challenges these notions but from a different direction. The complex products and systems literature questions the applicability of modular product architectures on the grounds of specificity of product and system designs (Miller et al. 1995), the prominent role of architectural innovation (Henderson & Clark 1990; Hobday 1998) and interdependencies among different design hierarchies (Murmann & Frenken 2006). Under competitive mass-market conditions, products tend to evolve in conjunction with customers' conceptions towards modular design hierarchies and dominant design (Utterback & Abernathy 1975; Clark 1985), yet such dominant designs remain largely absent in the complex products and systems engineering industries (Miller et al. 1995), which produce complex and highly-specific products and systems in low volumes. The projects which engage in complex products and systems innovation often require expertise at the levels of components and product architectures (Henderson & Clark 1990; Hobday 1998) in order to manage the interdependencies among design hierarchies of inclusion and control (Murmann & Frenken 2006). The hierarchy of inclusion refers to the hierarchical and nested organisation of parts which constitute the physical embodiment of a product, whereas the hierarchy of control refers to the parts and functional logic which controls the operation and behaviour of that embodiment. The multifaceted interdependencies among of the hierarchies of inclusion and control constrain the combinability of components and call for detailed knowledge of the system architectures as well as the functioning of different components (Prencipe 2000; Lee & Berente 2012). Against this backdrop, complex products and systems literature questions to what extent innovative work can be partitioned and coordinated through the processes of modularisation and modular product architectures.

As discussed above, although modularisation and modular product architectures are widely used in industrial settings and studied by academic

researchers, the extent to which these notions reflect the organising logic of digital innovation and complex products and systems innovation has been questioned, albeit on different grounds. Digital innovation literature does away with centralised design agency by turning into emergent and generative combinations, whereas complex products and systems innovation literature challenges the extent to which any modularisation or separation of concerns is feasible at all due to functional interdependencies that require detailed knowledge of product specific components and architectures.

This leaves organisations and researchers with a conceptual conundrum: which theoretical or conceptual logic or framework should they follow or use when thinking of and working on robots and autonomous systems? Should they draw the lessons from the domain of digital, or should the lessons learned from the domain of complex products offer a better starting point? This is not an easy question to answer since robots and autonomous systems can be viewed at the same time through the lenses of digital and complex innovation and be labelled as complex digital innovation. First, they are complex compositions of diverse sets of components which in combination are expected produce context-dependent and purposeful behaviour (Siciliano & Khatib 2008) and second, their behaviour is to a large extent controlled through the digital means, the computers and algorithms (Russell & Norvig 2010; Bekey 2005).

The purpose of robots and autonomous systems is to perform tasks with limited human intervention. To loosen the coupling between the machines and their human operators, the machines are equipped with a set of technologies that allow them to steer and control their own behaviour. These sets of technologies consist of sensors that gather inputs from the surrounding environment, control systems that convert environmental inputs into plans actions and actuation mechanisms which then exert actions back to the environment (Bekey 2005). Therefore, such machines, once set in motion, are expected to operate towards given goals by responding to and absorbing a variety of contingencies that materialise in the surrounding environment and are within the bounds of their control systems (Wiener 1965; Ashby 1958). In this light, the notion of autonomy should not be considered as an essentialist character that could be attributed to a particular machine. Instead, in the context of robots and

autonomous systems, the notion of autonomy is better understood as non-voluntarist situated actions, as a contingent behaviour which emerges from a machine's interaction with its surrounding environment. In this view, the degree of autonomy a machine possesses can only be evaluated in relation to the degree of contextual variety the machine is able to handle while pursuing a given goal in a specific environment.

Much of the progress in the field of robots and autonomous systems derives from the development and application of a variety of computing technologies and methods and across different domains of digital innovation. Large and small companies alike are confronted with the task of making sense of multidimensional and changing technological environment and considering how to conceptualise and arrange their innovative practices that revolve around digital, physical and autonomous technologies.

But therein lies the problem. The development of robust and reliable behavioural models leaves little room for uncertainty, requiring orderly and knowledgeable integration of multiple technologies, yet, as per the organising logic of digital innovation, knowledge and control can be highly distributed among the actors and organisations that participate in the development of digital artefacts. Therefore, the organising logic of complex digital innovation appears to reside uncomfortably between the two somewhat contradicting logics of combination, thereby prompting empirical investigations to get a better grasp of the problem.

1.1.4 Empirical observations

Over the past decade, digital innovation research has studied structural arrangements, organising logic (Yoo et al. 2010) and competition dynamics (Karhu et al. 2014) in the context of digital infrastructures (Tilson et al. 2010), platforms (de Reuver et al. 2017) and associated boundary resources (Eaton et al. 2015; Ghazawneh & Henfridsson 2012). This body of literature documents the convergence towards generative platforms and ecosystems that centre on certain operating systems and digital services.

Whereas platformisation and convergence characterise digital innovation in the context of desktop computing, mobile devices and digital business models in general, the prevailing state of affairs appears more diverse in the domain of complex digital innovation, namely, robots and autonomous systems. A cursory search reveals a few dozen proprietary and open-source platforms, software development environments and framework, which are aimed to support the development of robots and autonomous systems. Regardless of the apparent similarities in the architectural principles and functionalities they offer, they differ in implementation details and are not compatible with each other (Kramer & Scheutz 2006; Iñigo-Blasco et al. 2012). The naming of YARP, *Yet Another Robot Development Platform*, is presumably symptomatic of the situation. In general, endeavours such as YARP aim at improving the combinability and reusability of robotics-specific software to leverage previous efforts, yet the fragmentation and lack of interoperability among different components and sets of software are common. In addition, the manufacturers of industrial robots use proprietary software development environments and programming languages (Rossano et al. 2013; Pot et al. 2009). Moreover, advanced control systems are often embedded integrally to specific end-products, such as aeroplanes (Prencipe 2000) or self-driving cars (Thrun et al. 2006).

With various references to the benefits of platformisation, open standards and lessons from software engineering, there have been numerous attempts to make robot software more transferable in order to avoid reinventing the wheel. Regardless of the recent hype, it is worth noting that the first digitally controlled robotic arms were developed in the 1950s and put into industrial use in the early 1960s (Mason 2012). While many of the open-source efforts have emerged from the needs of developer communities, there have been corporation-driven efforts as well. For example, Microsoft launched Microsoft Robotics Developer Studio (MRDS) at the end of 2006 (Olsen 2006). The launch was accompanied by the article *A robot in every home* by Bill Gates (2007) in the Scientific American. The article pointed out the lack of common standards and platforms likening the state of robotics to that of computers in the mid-1970s. MRDS was then proposed as a way for consolidating contributions from the wider community. This would speed up the development of the industry by providing a common

set of standards and tools that cater the needs of the developers of robots and autonomous systems. Despite the good intentions, business opportunities did not materialise and Microsoft shut down the robotics group in autumn 2014 (Guizzo 2014). The generative promise of well-defined interfaces and standardised protocols remained elusive even for a well-resourced corporation.

Regardless of the apparent difficulty in establishing common standards, the Robot Operating System (ROS) (Quigley et al. 2009) has gained traction among researchers and developers in academia and industry. ROS is designed to support collaborative and distributed robot software development, and ROS and the related open-source community make a range of robotics-specific software freely available while bringing together roboticists and developers from different parts of the world. This distributedness renders the boundaries of knowledge and control diffuse uncertain and porous, which is something that could be considered as anathema in the view of design and engineering efforts that seek to produce reliable, predictable and robust systems whose behaviour is well-understood.

Therefore, the interesting part here is what makes ROS successful and how it manages to resolve the tensions between the contradicting organising logics that emerge from the specificity of designs and the distributedness of knowledge and control in complex digital innovation. Thereby, this tension provides the motivation and scope for this research.

1.2 Problem statement and research questions

The previous discussion presents an understanding of the tension emerging from the specificity of designs and the distributedness of knowledge and control in innovation literature. So far, this tension has received relatively little attention, and different bodies of literature offer contradicting views. This tension is restated below as a problematisation so as to inform the research question this work seeks to answer. The principal research question and two operative research questions are presented after the problematisation.

Robots and autonomous systems can be considered as complex digital innovation since they consist of a variety of physical and digital components,

and the development of goal-oriented, context-dependent and contingent behavioural models make extensive use of digital technologies. The literature on digital innovation theorise that novel products can be generatively combined from the components that originate from heterogeneous and distributed sources; by adhering to open standards, common interfaces and protocols, software developers can focus on their own areas of work with little regard to that what is hidden on the other side the interface. On the other hand, according to innovation literature on complex products and systems, designers and developers are expected to possess a detailed knowledge not only of the interfaces and particular components but the overall system architecture as well, due to multifaceted functional interdependencies among the hierarchies of inclusion and control. This brings up the tension between the specificity of designs and to the distributedness of knowledge and control of digital components. Furthermore, while the cursory evidence indicates the difficulty of establishing common standards in the field of robots and autonomous systems, the uptake of ROS shows that this is not entirely out of the question. These contradictions are problematic and raise a question on how complex digitised products should be conceptualised to make sense of their organising logic on innovation.

The motive of the principal research question rests on two pillars. The first one is the conceptual tension that emerges from the innovation literature, and the second is the existence and uptake of ROS, which gives a reason to assume the tension between the specificity of designs and the distributedness of knowledge and control can be resolved. However, how this is exactly done remains unclear. To this end, the following principal research question establishes the main objective of the inquiry:

How can the tension between the specificity of designs and the distributedness of knowledge and control be resolved in the development of complex and digitised products?

Next, the principal research question is restated as operative research questions to make the process of data collection and analysis more focused and tractable. Considering the exploratory character of this research, the operative research questions are left open-ended in order to retain the interpretative flexibility

while simultaneously directing the researcher's attention towards the potential areas of interest. The operative research questions are constructed upon tentative a priori concepts, which operate as sensitising devices and while remaining open in terms of what exactly should be observed.

The conceptualisation draws on Simon's (Simon 1962; Simon 1996) theory of hierarchies, which defines complex systems as nested, recursive and nearly-decomposable structures. Drawing on the central elements of this theory, the efforts of data collection and analysis are directed to the identification of subsystems and combination and their respective characteristics. They are expected to provide a well-rounded view on the organising logic of complex digital innovation in the context of ROS.

Starting from what is there to be combined, efforts are first geared towards the identification of the typical instances of subsystems at different levels of hierarchies of inclusion and control. Therefore, the first operative research question is formulated as follows:

What are the typical instances and characteristics of subsystems, if any?

Once the typical instances of subsystems are identified, the focus is shifted to the combination of subsystems with an emphasis to explore how they are combined across and at different levels of hierarchies. Therefore, the second operative research question is formulated as follows:

What are the typical instances and characteristics of combinations, if any?

The examination of the subsystems and their respective combinations is expected to provide a way to answer the principal research question and increase our understanding on how tensions between the specificity of designs and the distributedness of knowledge and control can be resolved in complex digital innovation.

1.3 Research objective and approach

As established in the principal research question, the objective is to study the tensions between the specificity of designs and the distributedness of knowledge and control. This is done with reference to ROS, which is widely used in the research and development of robots and autonomous systems. The research is exploratory aiming to provide a foundation for further studies. With reference to Gregor's (2006) categorisation of information systems theories, the aimed contribution can be characterised as type I theory, which seeks to describe and analyse the phenomenon. Such theories are described as follows:

"Says what it is. The theory does not extend beyond analysis and description. No causal relationships among phenomena are specified and no predictions are made" (Gregor 2006, p.620)

Theoretical contributions are presented in the form of proposed concepts (Eisenhardt 1989b), and whereas they may have a potential to shed light on where the future studies could be directed, no causal claims nor future predictions are made within the scope of this research.

Research design follows the framework presented by Eisenhardt (1989b). Data collection and analysis are guided by tentative a priori concepts, which direct the researcher's attention towards particular areas of interest. Furthermore, given the exploratory character of this research, data analysis and collection are partially overlapping so as to provide flexibility to pursue emerging avenues of research if and when data so suggests. The analysis of data is followed by the formulation of proposed concepts, which are then enfolded and discussed with reference to the literature on digital and complex systems and products innovation.

The emergence, organisation and use of ROS provide the empirical backdrop for this research, and a research database for ROS related data is constructed to serve two purposes. First, it provides a chronological trail of events based on which the case description is constructed. Second, to offer a rich body of evidence that enables an in-depth investigation for answering the primary research question. The research database consists primarily of documentary evidence, including ROS related blog entries, ROSCon conference presentations,

scientific papers and magazine articles and email archives. Together they cover a period from 2005 to 2016. The documentary evidence is supplemented with selected interviews and field notes.

The analysis of data proceeds in two stages. The first stage concerns with getting familiar with the context of research and case study. This stage produces the case description of ROS and develops an understanding of the language used in the field, and therefore serves as a preparatory stage before the second stage. The second stage focuses on a more detailed analysis of documentary evidence, using the method of thematic analysis to identify the themes and categories of subsystems and their respective characteristics of combination. This is done to describe and conceptualise the unfolding of the tension between the specificity of designs and distributedness of knowledge and control.

Subsequently, the proposed concepts are presented and discussed in the light of existing literature to evaluate their validity and the extent to which they are able to contribute to the literature on digital innovation.

The proposed concepts build upon the empirical evidence collected from ROS. Therefore, there are limitations to what extent the resulting conceptualisation applies to different robot software development environments. In addition, considering the complexity and open-endedness of ROS and the surrounding community, this research does not claim to be exhaustive. Therefore, there are presumably elements, which may contribute to the phenomenon but have not been considered in the analysis.

1.4 Targets for contributions

Should the objectives of this research be achieved, the following contributions to the innovation literature could be made.

The first area of contribution is the description and illustration of the empirical domain of robots and autonomous systems. This domain nor the software development environments or infrastructure that support it have not received much attention in the literature on digital innovation. Therefore, this work has an opportunity to contribute to the literature by introducing ROS, a widely used

software development framework for robotics and explicate its origins and organising principles.

The second area of potential contributions concerns conceptualisations which are expected to emerge from the thematic analysis. The thematic analysis is expected to reveal salient themes and properties that characterise the constitutive elements and organising logic of digital innovation in the context of ROS and in the development of robots and autonomous systems in general. This would contribute to the literature by providing conceptual tools that could potentially be of use when developing more targeted and fine-grained research designs to establish prevailing states of affairs at a greater level of detail and precision.

The third area of potential contributions is related to the innovation dynamics at the boundary of generative combinations and systems integration efforts. This is becoming increasingly relevant considering the broader uptake of technologies that are becoming more sophisticated, complex and autonomous. Therefore, developing a greater degree of understanding of the dynamics of complex digital innovation would presumably be also of practical interest for businesses and innovation and technology managers who are tasked to manage and increase the level of automation in different work environments.

Therefore, to aim at these contributions, this work proceeds to explore and examine how software development around ROS is organised, how it supports the innovation on robots and autonomous systems and what are the salient characteristics of the organising logic of innovation. The contributions of this research are discussed with reference to these objectives in the concluding chapter.

1.5 Structure for the dissertation

Moving forward, the next chapter, Chapter 2, reviews the related innovation literature in more detail. To this end, the focus of the review is on the body of research that deals with product architectures, modularisation, digital innovation as well as the characteristics of robots and autonomous systems and the innovation on complex systems and products. The problematisation and

principal research questions are presented after the literature review. The literature review is followed by Chapter 3 that introduces the theoretical framing that underpins the development of the tentative a priori conceptualisations and operative research questions. This is based on Herbert Simon's (1996; Simon 1962) theory of hierarchies which is concerned with the structural properties of complex systems. Chapter 4 describes the theoretical foundation of the research methods adopted in this research. Considering the exploratory character of this research, the methods applied seek to identify and explicate prominent themes that could be used to propose conceptualisations capable of informing the future research efforts. Chapter 5 presents the case description. The case description outlines the history and evolution of ROS and presents it as a framework and community that brings together a number of robotics researchers and developers from academia and industry. Then, Chapter 6 presents the results of the empirical study and brings forward the themes which emerged from the documentary evidence. Subsequently, building upon these themes, proposed conceptualisations regarding the complex digital innovation and the development of robots and autonomous systems are introduced. Subsequently, Chapter 7 discusses these proposed concepts in the light of existing literature on digital innovation and complex systems and products. The chapter begins by summarising the concepts and then proceeds to discuss the unfolding of the tensions between the integrality of designs and distributedness of knowledge and control in the context of complex digital innovation. Finally, Chapter 8 presents concluding remarks, which summarise this research by bringing forward potential contributions, outlining the various failings and limitations and suggesting avenues for future research.

2 Literature review

This research explores the organising logic of complex digital innovation. Given the range of dimensions along which organisations can innovate or support innovative practices, research on innovation ranges from human creativity, technological inventions, business and management practices to economic analysis and national strategies focusing on different levels and units of analysis (Ahmed & Shepherd 2010). In this work, the organising logic of complex digital innovation is studied in the context of robots and autonomous system.

Since the 1950s, the innovation that leverages digital technologies has played an increasingly important role in the process and product development (Nambisan et al. 2017). Information systems are widely used in organisational settings (Avgerou 2000), and digital technologies have come to constitute pervasive information and communication infrastructures (Tilson et al. 2010). Furthermore, physical and digital technologies are becoming increasingly intertwined and digital technologies manifest themselves in cars (Henfridsson et al. 2014), sensor networks (Sanfeliu et al. 2008), the Internet of Things and other cyber-physical systems (Broy et al. 2012; Conti et al. 2012; Wolf 2009).

Robots and autonomous systems represent the complex end of digital innovation. Typically, robots and autonomous systems are composed of a variety of physical and digital components in an attempt to create machines that are able to operate on their own by interacting with the surrounding environment (Bekey 2005). This is to a large extent facilitated by digital computing and artificial intelligence technologies (Russell & Norvig 2010).

This chapter reviews the related literature to establish the backdrop and boundaries for this research. Broadly speaking, the chapter unfolds along the five elements as illustrated in Figure 1. To begin, different types and definitions of innovation are outlined to locate this work in the wider body of innovation literature, and in the scope of this work, innovation is viewed through the lens of technological combination (Arthur 2009; Schumpeter 1983 first published in 1934). Then, the review directs its attention to the organising logics of innovation, which are viewed as the dynamics of combination in the context

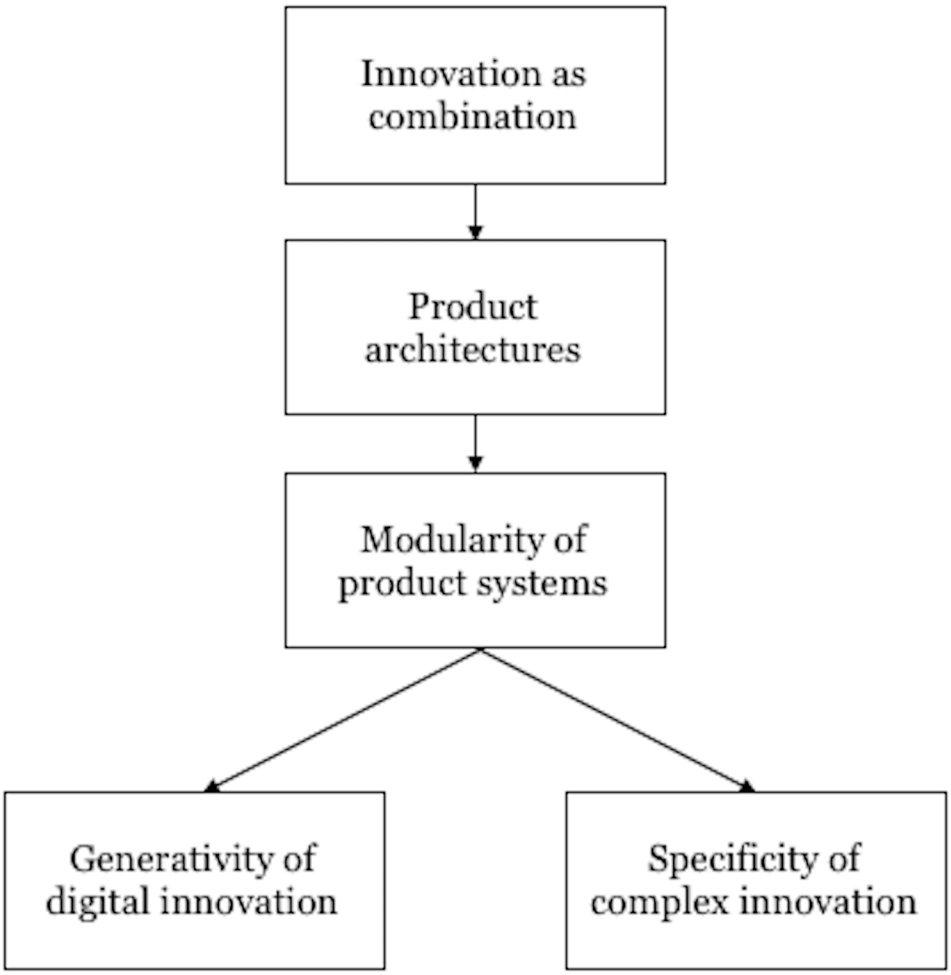

Figure 1: The structure of the literature review (own figure)

of product architectures. To this end, the concepts of product architectures (Ulrich 1995), modularisation (Baldwin & Clark 2000) and the modularity of product systems (Salvador 2007) are reviewed. After that, the two branches of innovation literature that challenge the modularity view are introduced and discussed. The branch on the generativity of digital innovation reviews the emerging body of digital innovation literature (Nambisan et al. 2017) with reference to software system architectures (Parnas 1972) and digital innovation as generative combination (Zittrain 2008; Zittrain 2006), looking also into digitised products in the view of layered modular architectures (Yoo et al. 2010) and complementary architectural frames (Henfridsson et al. 2014). After that, the branch on the specificity of complex innovation reviews the literature on complex systems and products innovation with reference to the inclusionary and control hierarchies (Murmann & Frenken 2006) and related implications to

the organising logic of innovation (Prencipe 2000; Lee & Berente 2012). At the end of the chapter, the reviewed literature is brought together to present the contradicting logics of combination, which leads to the problematisation and principal research question.

2.1 Innovation and novelty in technology

Innovation literature is varied and multifaceted, yet much of it derives from the economist Joseph Schumpeter's (1983) work published originally in 1934, which postulates that economic development emerges from novel combinations of existing resources.

The purpose of economic activity is to satisfy human wants and needs by producing goods. In turn, production is viewed as a combination of things and forces within our reach (Schumpeter 1983). Produces are then exchanged for money, and the flows of material and money circulate in opposite directions according to some established patterns of circulatory flows. Under normal conditions, these established patterns stay the same over economic periods as economic actors react routinely to changes in their outside environment. In contrast to the repeated cycles, innovation is an active attempt to alter some established pattern of material and monetary flows from-within the economy by combining existing resources in some novel way that is deemed advantageous by the markets (Schumpeter 1983). While the structural alteration of the material and respective monetary flows may prove highly beneficial to the innovators who seek to alter the flows to their advantage, these structural changes may spell a creative destruction for those who are not able to respond adequately.

The unfolding of innovation, the carrying out of novel combinations, can take place in a variety of ways and be analysed from a variety of viewpoints. In general, innovation is often characterised in terms of an idea or invention and its successful commercialisation (Ahmed & Shepherd 2010). It may result from a stepwise progress from an invention (technological novelty) to innovation (commercial adoption) and subsequent diffusion (wider uptake) which marks broader technological changes and shifts in business cycles (Ruttan 1959), yet inventing some technological novelty is not a necessary precondition for

innovation. Following Schumpeter (1983), innovation may unfold for example through the introduction of new goods or methods of production, the opening of a new market, the conquest of a new source of supply, or the carrying out a new organisation of an industry. This way, innovation also includes the transfer and application of existing technologies and knowledge to new product and application domains (Nambisan et al. 2017).

While Schumpeter's (1983) theory conceptualises innovation in terms of entrepreneurial effort to cause structural alterations in material and monetary flows, it remains silent on broader social and organisational processes and mechanisms that influence the unfolding of innovation (see e.g Ruttan 1959). To fill the void, a sizeable body of literature has emerged to examine different aspects of innovation at different levels and units of analysis (Crossan & Apaydin 2009), such as new product development (Garcia & Calantone 2002), process innovation (Davenport 1993) and service innovation (Vargo & Lusch 2004), or a variety of relationships among them (Utterback & Abernathy 1975; Crossan & Apaydin 2009), to provide a few examples.

Acknowledging the variety of definitions and research streams in the innovation literature, in this research, innovation is approached from the point of view that characterises innovation as a novelty in technology that is produced through the combination of existing technologies. This view builds upon W. Brian Arthur's (2009) work on technological innovation.

Whereas Schumpeter (1983) focuses on the entrepreneurial effort in carrying out novel combinations to alter the existing material and monetary flows, Arthur (2009) seeks to explain the role of technologies and technological evolution in this process. Furthermore, Arthur (2009) portrays economies as expressions of their technologies (of factors of production), and postulates that economies emerge and change in conjunction with the combinatorial evolution of technologies that produce them. In this view, innovation is seen as a novelty in technology and the process of technological innovation as combinatorial evolution in which new technologies arise through the combination of existing technologies.

Arthur (2009) describes technologies with reference to three fundamental principles. The first two of them focus on the logical structure, presenting technologies as a combination and as recursion. The combination principle, as illustrated in Figure 2, means that *"[a] technology is a combination of components to some purpose"* (Arthur 2009 Chapter 2), that is, technologies as systems are combined of components, such as subsystems, components and parts, which in combination produce some overall higher-level functionality. The principle of recursion, in turn, states that the components are technologies themselves. Defined this way, technologies are made of technologies, and can, therefore, be examined and evaluated as technologies across different technological domains, levels and units of analysis – and what is deemed to count as an overall system, subsystem or component is relative to the observer's objectives. These principles of combination and recursive organisation resonate with Herbert Simon's (1962; 1996) theory of hierarchy which presents complex systems as nearly decomposable and nested hierarchies.

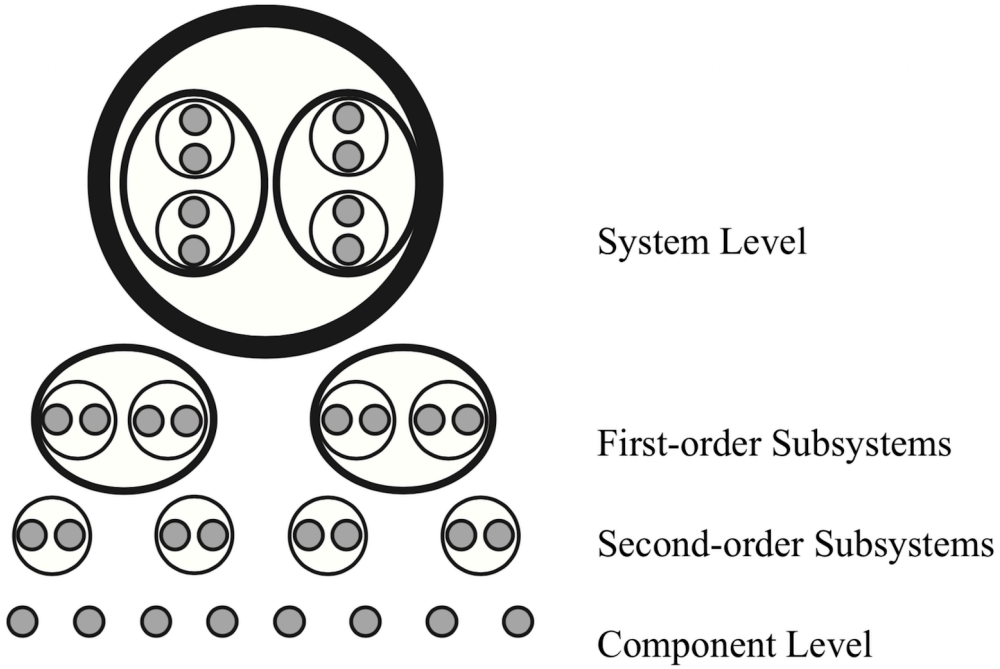

Figure 2: An illustration of a four-level nested hierarchy (Murmann & Frenken 2006)[1]

[1] Reprinted from Research Policy, 35(7), Johann Peter Murmann and Koen Frenken, Toward a systematic framework for research on dominant designs, technological innovations, and industrial change, Page 938, Copyright (2006), with permission from Elsevier.

The third principle turns attention from the structural to the functional logic, forwarding that all technologies harness and use some effect or phenomenon, or usually several (Arthur 2009). In order to perform desired functions, technologies incorporate some conceptual basic operational principle, a method or process that outlines steps and transformations that are needed to harness some natural, social or psychological effect or phenomena to serve some human purpose. More succinctly, technologies are purposed systems and the conceptual basic principle is *"the idea of an effect in action"* (Arthur 2007, p.274). For example, a digital camera captures patterns of light and turns them into digitally-encoded images (Yoo, Lyytinen, et al. 2010) that can be sent as bit-strings over digital infrastructures (Tilson et al. 2010) for people to share them on social media (Alaimo & Kallinikos 2017). The modern microprocessors and image sensors which make all this possible harness the semiconductivity of silicon in digital computation (Mack 2011).

Building on these definitions, novelty in technology is said to emerge from two primary sources, from the novel combinations of existing technologies and from the harnessing of effects or phenomena (which is a process that relies on existing technologies) (Arthur 2009). Depending on human and social goals, the processes of combination can vary from relatively trivial and gradual improvements to highly complex and experimental endeavours where newly harnessed effects and phenomena are put to use on a large scale.

Depending on the phenomena a technology harnesses, the principles of harnessing and the purposes they serve, technologies evolve and accumulate into different bodies and domains of technology. In turn, these reservoirs of technology can be drawn upon and combined to develop novel technologies, which in turn can be combined to create other novel technologies ad infinitum — changing constantly the backdrop against which new technologies and economic activities are performed. The gradual development and diffusion of new technologies, such as digital computation (Copeland 2014), may cause drastic shifts on how social (Kallinikos, Hasselbladh, et al. 2013), organisational (Tilson et al. 2010) and economic activities are arranged (Varian 2010).

To examine how logics of combination differ among different types and categories of innovation, the subsequent sections take a closer look into product architectures and review the related literature in the view of modularity, generativity and complexity.

2.2 Modularity and product architectures

This section reviews the literature on product architectures and modularity. First, the use of modularity in management literature is briefly outlined, after which product architectures and modularity are further discussed in the view of modularisation and centralised design agency.

Modularity is a general and widely used concept in organisation and management literature. It has come to carry a multiple of interpretations depending on the scale, context and phenomenon it is summoned to explain (Campagnolo & Camuffo 2009; Salvador 2007; Iman 2016). Although the notion of modularity can be conceptualised along different lines, it is commonly viewed as something that contains complexity and provides organisations with flexibility and agility through the separation and combination of concerns. Modularity allows organisations to respond more readily to varying market demands, competitive situations and technological change.

Campagnolo and Camuffo (2009) reviewed the use of the concept of modularity in management literature. Their findings reveal that management researchers interests in modularity revolve around the three areas of product design, production systems and organisation design. Product design, the first one of the three, had attracted the largest share of attention, and in this context, the concept of modularity deals with the allocation of functionality over components in product architectures and the implications of modularity on product lifecycles. The perspective of functional allocation draws upon Ulrich's (1995) conceptualisation of modularity as correspondence between functional elements and physical components. In turn, the life cycle perspective widens the view by taking into account different phases of a product life cycle, starting from design, production and use and ending with decommissioning. Second, in the context of product systems, research interests centre on the relationships among product modularity and outsourcing arrangements and their

implications on the formation of boundaries among firms and organisations (Schilling & Steensma 2001). Third, research into modularity in the context of organisation design seeks to explain the formation and adaptivity of organisational structures. The formation of organisational structures focuses on the existence of structural isomorphism between product architectures and development and manufacturing organisations (Brusoni & Prencipe 2006; Sanchez & Mahoney 1996; Baldwin & Clark 2000), whereas the adaptive view discusses how modularity of organisational structures provides flexibility that allows for agile responding to changes in market conditions (Ciborra 1996). This variation in definitions, measurements and contextual applications introduces ambiguity to the concept of modularity. Therefore, regardless of the apparent similarity of the underlying conceptualisation, reconciliation among different streams of modularity research remains challenging (Campagnolo & Camuffo 2009).

This diversity of views on modularity is also echoed also in literature reviews that focus on more narrow domains of research, such as service systems (Iman 2016) and product systems (Salvador 2007). The divergence of mechanisms which drive modularisation and subsequent combination in different socio-technical and technological contexts, scales and time frames appear to pose challenges to generalisation, regardless of the frequent calls for unified conceptual frameworks. The subsequent section discusses the concept of modularity in the light of product architectures.

2.2.1 Product architectures

Product structures are often discussed in terms of product architectures, and they can be characterised along the lines of integrality and modularity. A product architecture, following Ulrich's (1995) definition, refers to the *"scheme by which the functionality of a product is allocated to its physical components"* (ibid p. 419) – or, more precisely, can be seen a combination of *"the arrangement of functional elements, the mapping of functional elements to physical components and the specification of the interfaces among interacting physical components"* (ibid p. 420). In this view, functional elements are abstract and conceptual responses to functional requirements, which express the expected functioning of a product under some constraints and in a given

context. A mixture of functional elements and their interconnections are combined to produce a function structure which describes the overall functioning of a product; the function structure specifies the capabilities, constraints and behavioural logic of a product. Then, functional elements are allocated to physical components which provide the material substrate and produce specified functionalities as illustrated in Figure 3 (Ulrich 1995). Physical components can be viewed as separable parts which in combination produce the overall functioning of a product (Ulrich 1995).

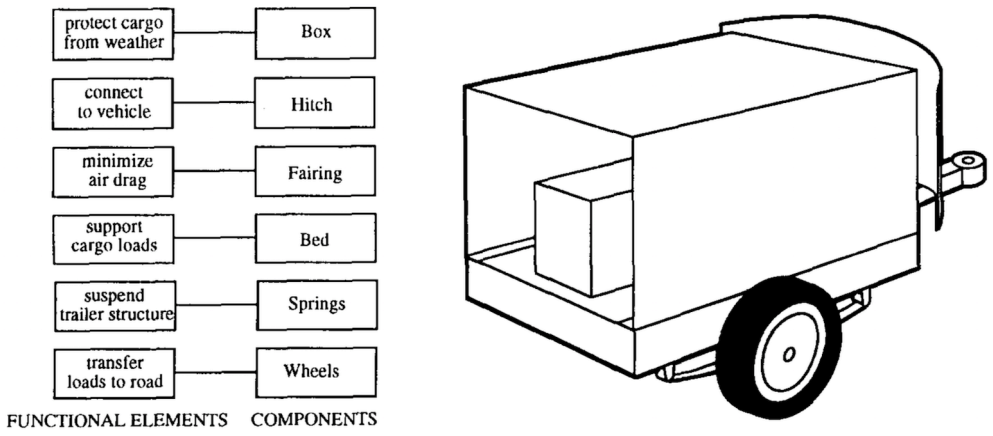

Figure 3: An illustration of functional allocation of a product architecture (Ulrich 1995)[2]

There are no definitive rules or guidelines on how functional elements should be allocated over physical components (Salvador 2007). Often, there are many different allocation schemes that could produce the same end result, that is, an equally functional end-product, and the eventual architecture result from design choices that are contingent on a variety of factors, ranging from the functional requirements to production systems and corporate strategies (Campagnolo & Camuffo 2009).

As the overall function structure is distributed over the components that constitute the end-product, individual components may come to perform a single function, be part of a composition of components which produces some single functionality or perform multiple functions simultaneously on their own (Ulrich 1995). A product architecture is called *modular* if there is a one-to-one

[2] Reprinted from Research Policy, 24(3), Karl Ulrich, The role of product architecture in the manufacturing firm, Page 421, Copyright (1995), with permission from Elsevier.

relationship between the functional elements and components and they are decoupled so that a change in one component does not require any change in the component(s) it connects to. On the other hand, product architectures are said to be *integral* if components implement multiple functional elements, and if components are coupled so that a change in one component requires changes in others. The detailed functional specifications of components depend on the allocation of functionality during the architectural design (Ulrich & Eppinger 2012).

When functional elements are distributed over a number of components, it is essential that the components are connected in a way that they produce the expected behaviour in combination. For this to happen, interfaces need to be located and specified in an appropriate manner. Otherwise components cannot connect to and interact with each other. To this end, interface specifications detail the necessary modalities of connection and interaction, such as protocols and physical coupling, depending on the types of connection and interaction (Ulrich 1995).

By allocating functional elements over components and establishing necessary interfaces, architectural design sets out the structural composition which produces the overall functionality of the product. As architectural design often resides at a higher level, it is complemented by more detailed specifications which provide lower-level details of different components and their respective interactions. These architectural schemes, blueprints and design documents allow for the communication of the overall vision among project teams (Hobday 1998) and facilitate the partitioning of work (Hippel 1990).

Product architectures, as any systemic arrangements, can be evaluated based on their structural patterns and respective properties (Bertalanffy 1968). The relationships among functional elements and physical components can be defined in terms of integrality and modularity (Ulrich 1995). However, product architectures are rarely fully modular or integral, and any attempt at categorisation depends on how and at what level of abstraction functional elements and physical components are defined and analysed (Salvador 2007). In this light, the matter of modularity and integrality should be considered

rather as a frame-dependent continuum between modularity and integrality than any general binary condition (Ulrich 1995; Yoo et al. 2010). While definitions vary, the notion of modularity is often used to refer to a degree to which components can be separated and subsequently combined (Schilling 2000).

2.2.2 Modularisation and task partitioning

As discussed above, product architectures vary along the spectrum of modularity and integrality (Ulrich 1995). At one end, a modular product architecture renders one-to-one mappings between functional elements and components, while, at the other end, an integral architecture produces intertwined and more interdependent relationships among different functional elements and components.

Modularisation is a design principle and process which seeks to decompose the overall design requirements and function structure in a manner which results in a modular product architecture (Baldwin & Clark 2000). The purpose of modularisation is to produce an overall structure, a product architecture, where modules have strong structural connections and interdependencies within the boundaries of a module, but the connections and interdependencies across the modules remain relatively weak. Subsequently, the modules which emerge from this directed and purposeful problem-solving process are often described as loosely-coupled (Schilling 2000; Parnas 1972). The loose coupling among modules brings several benefits, such as the commonality and combinability of components (Salvador 2007). The commonality allows modules to be used in different products, whereas the combinability of modules provides companies and customers with a larger variety of product variants in a cost-efficient manner since a large variety of products can be produced by arranging components in different configurations (Langlois & Robertson 1992; Schilling 2000). Moreover, companies can react to market changes in a more agile manner when they rely on modular and configurable product structure (Garud & Kumaraswamy 1995).

In addition, when design problems grow in size and complexity, it becomes increasingly challenging for any single individual or organisation to manage all

aspects of technological and design knowledge (Garud & Kumaraswamy 1995; Langlois & Robertson 1992). By modularising an overall design into partitions, work can be distributed over a larger group of people. This way, work can proceed in a parallel and directed fashion individually; partitioning a problem into a series of subproblems and distributing it over the specialised teams of designers and engineers, organisations can benefit from the division of labour as they can draw upon knowledge and skills of specialised workforce, increasing the speed of development and modular and architectural learning (Sanchez & Mahoney 1996). However, modularisation may not always lead to optimal product architectures, for example when the overall performance requirements are particularly high (Yoo et al. 2010; Ulrich 1995) or the synergistic specificity among modules bears influence to what extent the separability and subsequent combinability of modules is feasible (Schilling 2000).

Carrying out partitioning is costly and requires expertise and centralised planning (Baldwin & Clark 2000). To ensure that different parts work together once completed and combined, coordination among different stakeholders prior to partitioning is required (Ernst 2005; Danese & Filippini 2010). Ideally, task partitioning should follow the structure of the underlying design problem, yet it is not always straightforward to establish the relationship between the two (Alexander 1964), this especially being the case when problems are complex and multifaceted (Ethiraj & Levinthal 2004) or deviate from what went before (Sosa et al. 2004; Hippel 1990).

To coordinate work and manage complexity, Baldwin and Clark (2000) propose that connectivity among different modules is predicated on well-defined design rules. These design rules define the overall architecture of a product, interfaces and interconnections among modules along with the relevant integration protocols and testing standards. Clearly defined design rules enable the distribution of design and manufacturing tasks within and across organisations. This way, modularisation helps create embedded coordination mechanisms and information structures for organising and distributing work (Sanchez & Mahoney 1996).

2.2.3 Modularity as a property of product systems

Product modularity, that is, the separability and combinability of a range of components (Schilling & Steensma 2001), tends to occur under relatively well-defined circumstances. These circumstances often revolve around central platforms (Baldwin & Woodard 2008) and design rules that derive from the process of modularisation (Baldwin & Clark 2000) that seeks to achieve particular objectives through the modularity of a product, such as the cost-effective production of a variety of products or their serviceability and recyclability. To these ends, modularisation partitions the overall design in a centralised top-down manner.

Considering that modularity often seeks to produce product variety through the combination of different but compatible components, modularity can be better understood as a property of some particular product system than as a property of an individual product or any general combinability (Salvador 2007). A modular product architecture describes the relationships, the degree of variation and the combinability of components within the boundaries of a product system where modules represent units of variation (Salvador 2007).

In addition, Salvador (2007) posits that combinability and product variety occur when the following conditions are fulfilled. First, a module that produces variation is internally cohesive so as to produce the required variation independent of other components. Second, the interface design parameters that define connection and interaction between components are tightly coupled, as otherwise modules could not be connected. Third, to produce variation, a certain degree of freedom is allowed for component designers in terms of non-interface design parameters. However, coming up with specifications for all three is often challenging given the ambiguity and differing objectives of the schemes that are used to capture and allocate different functional aspects to modules. Moreover, while modular function allocation and specification can be designed within the bounds of a product system architecture whose overall requirements are known in advance, designing such bindings is a precarious effort when functional requirements are not fully known.

The overall architectural design of a product system is expected to specify design parameters that outline opportunities and limitations for modularisation in different scenarios. While this kind of arrangement is able to produce variety, it does so along the lines of well-defined design hierarchies and corresponding market concepts (Clark 1985) that leave little room for serendipitous emergence of technological novelty. Indeed, much of the literature on modularity draws its lessons from the industrial manufacturing of mass-market products, such as cars (Clark 1985), consumer electronics (Langlois & Robertson 1992) or semiconductors (Baldwin & Clark 2000), where the capability to produce cost-effective product variation and change are considered as of strategic importance (Garud & Kumaraswamy 1995).

Occasionally, as products and markets mature, the focus of innovation shifts from product innovation to process innovation (Abernathy & Utterback 1978). This crystallisation of product and market concepts and how they are best achieved give rise to dominant designs and design hierarchies (Clark 1985), which come to dominate the industry-wide organisation of firms and markets (Suarez 2004; Murmann & Frenken 2006). This brings fixity and commodified competition to technological arrangements which provide a backdrop against which technological development and economic activities unfold (Arthur 2009).

As a final note, it is also worth noting that the modularity of a product system is not invariably the same as the partitioning of design or production (Salvador 2007). Design work can be partitioned and distributed across multiple designers even if the structure of the end-product would be highly integral. On a similar note, the production of large structures, such as cruise ships, can be performed in a piecemeal manner, yet this does not indicate that the end-product would be modular in the view of the separability and combinability of components in the context of a product system.

To summarise, the concept of modularity in the contexts of product architectures builds upon the underlying assumption of a centralised design agency that carries out the partition of the overall design in a top-down manner. The literature on digital innovation challenges this assumption and argues that

digital products and services emerge as generative combinations through decentralised design agency. This is discussed in the subsequent section.

2.3 Generativity and digital innovation

The research into digital innovation maintains that the pervasive and increasing use of digital computers and advanced information processing technologies enables novel product architectures and organisational arrangements (Nambisan et al. 2017), which in turn are prone to generate profound changes in firms' organising logic and innovation management practices (Yoo et al. 2010; Tilson et al. 2010; Henfridsson et al. 2014; Barrett et al. 2015). Traditionally, information systems research has focused its efforts on communication, transaction and decision-support systems in organisational contexts (Avgerou 2000), yet the increasing digitalisation across application domains and social contexts suggest that a broader lens should be adopted (Grover & Lyytinen 2015; Nambisan et al. 2017). To this end, Fichman, Dos Santos and Zheng (2014, p.329) forward *"digital innovation as a fundamental and powerful concept in the information systems curriculum"* while others argue that digital innovation and information systems research could serve as a reference discipline to product and service innovation management fields (Yoo 2012b).

Digital innovation revolves around the processes of digitisation and digitalisation (Tilson et al. 2010). Digitisation as a technical process refers to the development and application of techniques and technologies that render and manipulate objects and artefacts in the format of binary digits. Digitalisation, on the other hand, refers to the dynamic interplay of social and technical matters, which are related to the wider adoption and deployment of digitisation techniques in social and institutional contexts. Out of these two, the main focus of digital innovation research is on the socio-technical matters of digitalisation, whereas the socio-technical and organisational outcomes of digital innovation are often explained fully or partly in terms of digitisation.

Considering that digitisation and the application of digital technologies are often regarded as factors that explain the unfolding and outcomes of socio-technical innovation, the following subsection takes first a closer look into the fundamental characteristics of digital computation, after which the notion of

modularity in the context of computer systems and the generative properties of digital technologies are discussed further.

2.3.1 Digital computation

In the 19th century, people as human computers created tables of precomputed values for many mathematical problems (Grier 2005). As the tables often contained errors, in the 1820s Charles Babbage started developing mechanical computers that would supersede their human competition in terms of accuracy and speed (Swade 2002). While Babbage's creations, difference and analytical engines, never really took off, they are often regarded as early ancestors of modern computers as they rely on mechanical switches that carry out calculations and store intermediate results. About a hundred years later, in 1936, Alan Turing introduced a theoretical device to conceptualise the functioning logic of a universal computing machine (Copeland 2014). The Universal Turing Machine conceptualises the process of computing as a series of arranged and stepwise instructions that operate on and manipulate symbolic representations, which are read from and stored in memory (tape) one step at a time. A few years later, in 1945, John von Neumann, forwarded the concept of stored program computers (Copeland 2014). With programs (instructions) represented and stored in memory along with data, the functioning of a computer could be changed by equipping it with a new set of instructions, thereby leaving the fixity of earlier computer designs behind and making them readily applicable to different purposes. As both data and instructions are commonly encoded and stored in the format of binary digits (bits), they can be transferred across and processed on different computers as long as they follow the same logic of functioning, at least in principle if not always in practice. The progress from mechanical switches to ever smaller, cheaper and ever more powerful microprocessors (integrated circuits) (Mack 2011) has paved the way to increasingly pervasive and ubiquitous application of computers in various tasks and domains (Yoo et al. 2010).

Although the first computers were intended to carry out numerical calculations, the principles of logical symbol manipulation can be applied to a variety of information processing purposes (Copeland 2014). Modern digital microprocessors (or more generally, digital computers) are programmable

general-purpose machines; in principle, they are able to process any set of data as long as both the data and the stepwise instructions to process it are represented in an unambiguous format, are congruent and effectively computable (Tilson et al. 2010). If an object or artefact can be encoded and represented in an unambiguous format of binary digits, and a set of logical operations (stepwise instructions/algorithms) can be produced to carry out necessary operations upon it, any matter may become computable. No matter if it deals with fake news, differential equations, numbers and letters on the screen, maps, long-winded decision trees, missile trajectories or data transmission protocols and procedures. However, the intractability of some computational problems and limits in memory and processing capabilities render some classes of computations and tasks practically infeasible (np-hard), although advances in processor and memory technologies (Mack 2011) and computing techniques (Russell & Norvig 2010) are pushing the boundaries of computability and thereby opening up constantly new opportunities for digitalisation. Moreover, as both data and instructions are stored in a computer's memory, from which they are transferred to the processor a computing cycle at a time, both data and instructions are volatile and can be changed with relative ease and minimal material resistance using other digital means without modifying the material substrate that carries out the computation (Kallinikos 2012; Kallinikos, Aaltonen, et al. 2013).

The purpose of a processor is therefore contingent upon the task and instructions it carries (out) at any particular moment, or, as Arthur (2009) puts it, microprocessors are minuscule information processing factories, factors of production, that can be furnished and arranged for a variety of more or less productive purposes. Therefore, the digital computer provides a universal method to carry out the manipulation of symbols (bits). Moreover, as both the symbols and the instructions to manipulate them can represent a variety of social and natural objects, artefacts and processes, the computer can be considered as a general-purpose technology. However, as the computation of a particular computer has to be well-defined, the instantiation of a general-purpose machine turns it to a special-purpose machine, albeit a fickle one.

This way, the underlying characteristics of digital innovation can be traced back to bitwise representations of data and instructions and the stored program (von Neumann) architecture of modern computers. In this view, Yoo et al. (2012) conceptualise the fundamental properties of digital technology as homogenisation of data, reprogrammability, which drive distributed and combinatorial innovation. On a similar note, Kallinikos et al. (2013) define digital artefacts as editable, interactive, reprogrammable and distributable objects, which undergo constant change and transfiguration as multiple stakeholders in broader digital ecosystems strive to use and modify them so that they serve their particular purposes. Also, given their reprogrammability, digital artefacts are described as incomplete, open-ended and continually in the making (Garud et al. 2008), and a source for generative combinations and procrastinated binding (Yoo 2012a). This malleable and mutable character of digital artefacts questions the assumptions of stability, provenance and control frequently associated with physical items and devices (Yoo et al. 2010; Garud et al. 2008; Kallinikos, Aaltonen, et al. 2013; Lyyra & Koskinen 2016). These malleable and mutable characteristics of digital objects and artefacts are occasionally referred to as digital materiality (Barrett et al. 2012; Yoo et al. 2012) to differentiate from the more tangible and inertial qualities of physical materiality.

These foundational characteristics of digital computation and computers are often cited as a partial explanation of technical, social organisational outcomes and arrangements in digital innovation at different levels, while simultaneously fully acknowledging the role of social and institutional factors as well. Research into digital infrastructures (Tilson et al. 2010), platforms (de Reuver et al. 2017), digitised and connected products (Henfridsson et al. 2014), automation and virtualisation of work (Bailey et al. 2012) have brought forward different aspects of digitalisation in various socio-technical settings.

2.3.2 Modularity in computer systems

Digital computation provides a general method for performing symbol manipulations in a highly flexible and malleable manner (Copeland 2014). However, the designing and developing of computing systems require coordination and organisation of human efforts (Baldwin & Clark 2000).

Similarly to other products, digital products can also be subjected to modularisation at various levels in order to subdivide design and development tasks and to manage complexity.

At the level of physical products and computing machinery, for example, the desktop computer can be considered to follow a modular product architecture (Garud & Kumaraswamy 1995; Langlois & Robertson 1992). A desktop setup normally consists of a central processing unit, display, keyboard and mouse, and perhaps a printer and some other peripheral devices, and these modules operate in combination and interact with each other through well-defined interfaces. As long as connections and modalities of interaction conform to particular interface specifications, manufacturers and users are able to assemble different product configurations from a diverse range of modules to have a configuration that best serves their needs.

Modularisation is a common approach in software development as well. A variety of approaches can be used to carry out the partitioning of design problems, yet some of them are considered to be more beneficial than others. Parnas (1972) forwards an idea of approaching the partitioning in terms of responsibility assignment. In other words, a software component, a module, should not simply correspond to some process step in the program flow, but to the functionality it provides. This way, by allocating a module with some well-defined function and selecting an appropriate level of abstraction and interface definition, the knowledge of underlying design decisions which produce the functionality of the module can remain hidden. This is commonly known as the principle of information hiding, and it facilitates separation of concerns while providing a scheme based on which work can be distributed among software developers; large software projects are routinely partitioned into smaller chunks in order to reduce complexity to manageable levels. Typically, such effort begins by detailing the expected functionality of a product, after which the overall architecture is designed. Subsequently, the architecture details the structure of the resulting software in terms of its components, their expected functionality and respective interconnections, thereby providing an overall scheme based on which design and development work can be distributed among software developers (Baldwin & Clark 2000). This decomposition through the separation

of concerns allows for a more piecemeal and parallel approach in development, testing and maintenance, while also enabling the reuse of software components. This way, modular architectures facilitate separation of concerns and enable parallel and distributed development, which leverages specialised knowledge (Sanchez & Mahoney 1996). For example, mobile device application developers need not be aware of the fine details on wireless message routing when developing software for consumers. Also, changes in the system functionality are easier to implement when the interdependencies between components are lower, containing code changes to a smaller number of components (Parnas 1972).

Furthermore, separation of concerns is also applicable to large-scale information and communication infrastructures and systems that operate across organisational boundaries. For example, Open Systems Interconnection (OSI) model[3] conceptualises the architecture of computerised communication systems as a layered stack. The OSI stack is an abstract conceptualisation, a reference model, which describes the expected functionality of components that reside at different layers of functional hierarchy, detailing characteristics and expectations at the levels of physical connections, data linking, network routing, data package transport, session management, data presentation and services to applications. As a reference model, its goal is to facilitate comparability between different communication protocols by characterising the essential qualities and features of different layers and their respective functioning. At the same time, it provides a paradigmatic example of the layered architecture of digital technologies. In practice, there are several implementations and variations of the OSI model. Using well-received protocols and standards which adhere to a layered structure, such as TCP/IP, a variety of communication systems can be constructed generatively from different components.

2.3.3 Generative combinations

The layered aspect of digital product architectures is postulated to provide the foundation for the generative properties of digital ecosystems (Yoo et al. 2010) and infrastructures (Tilson et al. 2010). The concept of generativity has its roots

[3] wikipedia.org/wiki/OSI_model

in Jonathan Zittrain's (2008; 2006) work, which seeks to explain the proliferation and evolution of the Internet.

The notion of generativity builds on the observation that digital networks and artefacts are arranged as layered stacks (Zittrain 2008), such as communication networks and operating systems, which can be connected with each other through open standards and interfaces that serve as gateways between different layers. This layered structure, which can also be referred to as an hourglass architecture (Zittrain 2008) allows for task partitioning, separation of concerns and information hiding (Parnas 1972), thereby enabling the participation of wider audiences by lowering the threshold of participation (Zittrain 2008). For example, digital content and data can be transmitted across the Internet using the Internet Protocol without needing to know how to route messages between senders and receivers. In a similar fashion, application software can be created and shared over the Internet while common operating systems provide a foundation upon which developers and users can build and run their applications. Against this backdrop, Zittrain (2008) conceptualises this open and layered quality of the Internet as generativity and summarises it as follows: *"Generativity is a system's capacity to produce unanticipated change through unfiltered contributions from broad and varied audiences"* (Zittrain 2008, p.70).

To elaborate further, Zittrain (2008) outlines five characteristics of technologies that empower audiences and enable their contributions. These characteristics are leverage, adaptability, ease of mastery, accessibility and transferability. To begin, leverage helps achieve results with lesser efforts. Adaptability, in turn, indicates the level of effort needed to build upon a technology, or to modify and broaden its range of uses. Ease of mastery indicates the steepness of the learning curve before one becomes knowledgeable and skilled enough to master the technology in question. Accessibility means simply whether potential users have access to technology, tools and documentation, whereas transferability indicates how readily users' contributions can be conveyed and shared with others, possibly less-skilled users. These five characteristics constitute generativity: *"The more that the five qualities are maximized, the easier it is for*

a system or platform to welcome contributions from outsiders as well as insiders" (Zittrain 2008, p.74).

Technologies differ in terms of their above-mentioned characteristics, and the absence of any of these characteristics may render a particular technology less generative (Zittrain 2008). For example, the Linux kernel is generative in terms that it provides leverage, adaptability, accessibility and transferability, yet the difficulty of mastery tends to keep contributions from broad and varied audiences rather limited, at least when compared to the creation of static web-pages with simple hypertext mark-up language. In addition, generativity may manifest itself at different technological layers. For example, the generative pattern of Wikipedia content creation does not necessarily mean that the underlying technological layers which facilitate the generative creation of content are generative themselves, although they often are (Zittrain 2008). Zittrain (2008) also differentiates between generative tools and generative systems. Generative tools are useful on their own and in individual terms. In turn, the notion of a generative system refers to larger technological arrangements and can be viewed as *"a set of tools and practices developed among larger groups of people"* (Zittrain 2008, p.74).

The generative characteristics of systems lay out foundations and conditions which allow for innovation and novelty to emerge from the grassroots through the contributions from wider audiences (Zittrain 2008), standing thereby in contrast to the centralised top-down planning and coordination which characterises modularisation through decomposition (Yoo 2012b). This way, the process in which components are created and consumed changes its direction. Whereas the process of modularisation allocates modules with specific functionalities when an overall design is subdivided over different modules, with generativity components originate from different sources and without a centralised design agency. This way, a component remains agnostic in relation to an end-product at the time when it is designed and created, as it receives its final purpose and meaning with respect to a particular end product at the time it is included into a combination that constitutes the end-product. The situation in which the allocation of functionality to a component takes place after the

completion of the design of the component can be referred to as procrastinated binding (Yoo 2012a).

End-product agnostic design, procrastinated binding, recombination and the associated generative and emergent characteristics of innovation result in dynamic complexity (Hanseth & Lyytinen 2010) and paradoxes of change and control (Tilson et al. 2010). For example, while the technology and internet companies may provide and open up some particular boundary resources, such as system development toolkits, interfaces and provide access to computation in order to attract wider participation, they also seek to guide the trajectories of development to their benefit by controlling the pathways through which the audiences are allowed to participate (Ghazawneh & Henfridsson 2012; Eaton et al. 2015).

2.4 Digitised products

The increasing digitisation of tangible products has inspired researchers to re-examine the current conceptualisations of product architectures. To address the interplay among the digital and physical aspects of products, recent research (Yoo et al. 2010; Henfridsson et al. 2014) has developed conceptualisations to take into account the differing levels of resistance and readiness for change among the physical and digital components, and demonstrated the subsequent implications on product design practices, product architectures and lifecycles.

This stream of research highlights the differing levels of material inertia and resistance to change among the digital and physical components that constitute digitised products (Barrett et al. 2012) and characterises this in terms of digital and physical materiality (Yoo et al. 2012). The differing levels of material resistance can be traced back to the reprogrammable character of the digital computer (Tilson et al. 2010; Henfridsson et al. 2014). In the end, the digital computer is a general-purpose machine (Copeland 2014), which can be repurposed by supplying it with a new set of instructions. While it may take significant time and effort to develop the first set of instructions, however, once the set has been created, the marginal cost of its distribution and installation on multiple computers is negligible. On the contrary, physical components lack the mechanism of cost-efficient change en masse, and each repurposing of a

physical component requires resources, such as raw materials, tools and labour, to carry out the required changes. Given the higher marginal cost associated to the repurposing of physical components and products, product manufacturers seek to specify product features and verify the correctness of product designs with a great care before transferring products from design to manufacturing (Henfridsson et al. 2014). Therefore, the physical components of a product rarely undergo change once they have been manufactured, whereas the product functionality which is implemented and controlled using digital means may undergo frequent change throughout the lifecycle of a product (Lyyra & Koskinen 2016).

This challenges the prevalent views on product and software design and engineering. Conceptual challenges revolve around the extent to which the traditional views and practices are able to capture the intertwined character of differing levels of material resistance and readiness for change during the design and manufacturing processes (Henfridsson et al. 2014) and throughout the product lifetime (Lyyra & Koskinen 2016).

To conceptualise the combination of the digital and physical components, the layered modular architecture (Yoo et al. 2010) shows how adding digital components into traditionally fixed and single-purpose products can transform them into platforms which are open for modification and repurposing long after the physical embodiment of a product has received its final form. To prepare for this, Henfridsson et al. (2014) forward that product designers and architects should adopt the notion of complementary design frames to conceptualise product architectures which consist of overlapping digital and physical components and that are developed and evolve at different speeds.

2.4.1 Layered modularity architecture

With the notion of layered modular architecture, Yoo et al. (2010) separate product innovation from process innovation and maintain that the digitisation of products is altering product architectures, and by extension the organising logic and innovation practices of organisations (Sambamurthy & Zmud 2000). To conceptualise the incorporation of digital technologies into physical products, Yoo et al. (2010) bring together two architectural schemes, the layered

architecture of computer systems and the modularised architecture of product systems. This amalgamation is then referred to as the layered modular architecture.

The layered modular architecture "*extends the modular architecture of physical products by incorporating four loosely coupled layers of devices, networks, services, and contents created by digital technology"* (Yoo et al. 2010, p.724). The layered architecture forms a hybrid architecture where the *"degree by which the layered architecture adds the generativity to the modular architecture forms a continuum"* (ibid p. 728). To exemplify, *"[t]raditional industrial-age, single-purpose products manifest one end of the spectrum while conventional digital products with general computer hardware form another end. Many digitized products will fall somewhere in the middle"* (ibid p. 729). In this light, the architecture and lifecycle of a product consist of two elements. First, the design and manufacturing process during which the physical embodiment of a product receives its final shape and, second, the more open-ended and generative phase as the product and its functionality can be changed throughout its lifecycle through software changes. The layered modular architecture conceptualises digitised products as generative platforms by showing how layered and modular architectural schemes unfold during the product lifecycle (Yoo et al. 2010).

Traditionally, the notion of modularity, which stands at one end of the spectrum, is tightly linked with the idea of modularisation (Baldwin & Clark 2000) in the view of the product architectures (Ulrich 1995), product systems (Salvador 2007) and design hierarchies (Clark 1985). Modularisation starts by forming a complete plan of a product or a product system, after which the complete plan is partitioned into smaller modules according to certain objectives and rules (Baldwin & Clark 2000), for example with an aim to produce a variety of product configurations while minimising costs (Schilling 2000) or to leverage division of labour in design and manufacturing (Sanchez & Mahoney 1996). Such products are not expected to change once they are manufactured. However, the incorporation of digital components adds a more open-ended logic into physical machinery.

The concept of the layered modular architecture adds the layered stack architecture of software on the top of the modular architecture that forms the device layer (Yoo et al. 2010). This is illustrated in Figure 4. At the bottom of the stack resides the device layer, which consists of physical hardware and computing machinery as well as an operating system which provides the logical layer of abstraction and modulates the interaction between the computing machinery and upper layers of the stack. On the top of the device layer is the network layer. The network layer consists of a physical transport media (e.g. antenna, cable) and logical transmission protocols (e.g. TCP/IP) and it is used to connect to other computers and digitised products. Then, on the top of the network layer is the service layer where different applications and their respective functionality reside, above which, at the top of the stack, is the contents layer which holds data.

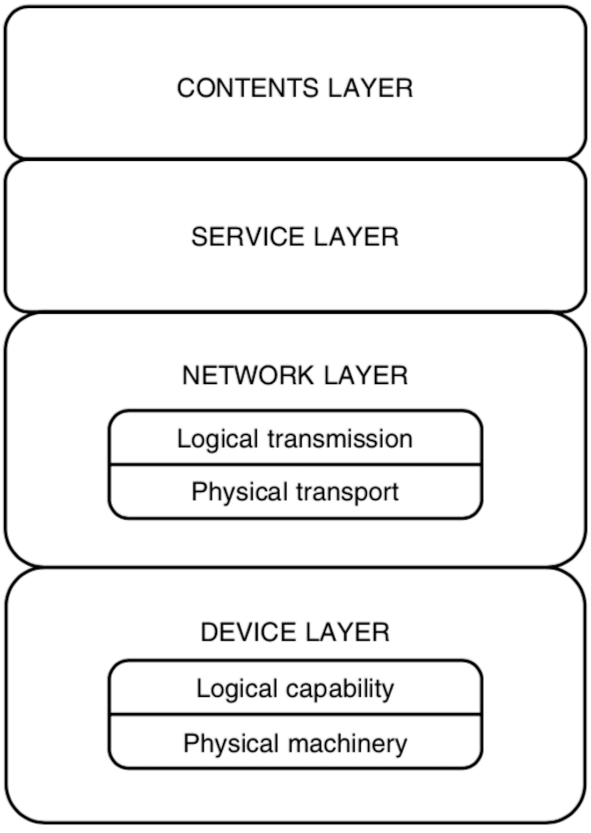

Figure 4: The layered architecture of digital technology (Yoo et al. 2010) [4]

[4] Republished with permission of Information Systems Research, from The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research, Youngjin Yoo, Ola Henfridsson and Kalle Lyytinen, 21/4, 2010; permission conveyed through Copyright Clearance Center, Inc.

This way, the underlying hardware and operating system come to provide a stable platform upon which novel digital functionalities and services can be developed in a generative manner (Zittrain 2008), within the bounds of the physical characteristics of the product (Yoo et al. 2010). Therefore, whereas hardware goes through the processes of modularisation and production similarly to other manufactured products at the beginning of a product lifecycle, the digital part of the product remains open-ended and can be improved throughout the lifecycle.

Platforms and open interfaces allow for wider audiences to take part in innovation as they can develop and provision applications, services and content that can be used and consumed on different platforms, such as on iOS devices (Eaton et al. 2015). Users can select and install applications from application stores and use services that best serve their needs, rendering the devices that adhere to the layered modularity as enacted ensembles (Yoo et al. 2010) which undergo a continuous change as new applications, services and content are made available and taken into use. Furthermore, it is worth to note that digital platforms are not necessarily directly linked to some specific hardware and devices but can also operate as independent cloud-based services (de Reuver et al. 2017) – Google Maps provides a paradigmatic example of this (Yoo et al. 2010).

Whereas modular product systems produce a degree of variety within the confines of premeditated product designs (Salvador 2007) and design hierarchies (Clark 1985) which are nested and fixed, the layered modularity facilitates the combination of end-product agnostic components, applications, contents and services from different design hierarchies as long as they adhere to certain technical standards and boundary resources (Eaton et al. 2015). In this light, the digitisation of products tends to render design agency more distributed (Nambisan et al. 2017). While this generative character provides a broader repertoire of applications and services, it also indicates that no single party is able to fully control digitised products and their configuration (Yoo et al. 2010).

The organising logic of the layered modular architecture can be viewed as doubly-distributed since the control and knowledge of components and combinations are distributed across heterogeneous groups of actors and organisations (Yoo et al. 2010). Learning and mastering the ways to cultivate and harness distributed innovation among heterogenous audiences is vital to the success of organisations that pursue platform-based product strategies.

2.4.2 Materiality and product architectures

The differing levels of the material resistance to change between the digital and physical components bear implications across different phases of a product lifecycle, ranging from product design and manufacturing to the phases of use and retirement. This dissociation of physical and digital materialities shapes product design practices and dominant designs (Hylving & Schultze 2013).

As discussed in the previous section, incorporating digital components into traditional physical single-purpose products and opening them up for broader audiences to take part in innovation may turn them into generative platforms that evolve throughout the lifecycle of a product (Yoo et al. 2010). However, the digitisation of a product does not automatically imply that the manufacturer opens it up as a platform to attract third-party developers. For example, Tesla frequently updates the functionality of Tesla cars by delivering software updates which add new functionality or improve the existing ones (Lyyra & Koskinen 2016), without providing access to third-party developers.

In either case, product designers and manufacturers are advised to take the differing levels of material inertia into account during the architectural design. Based on the empirical observations from the design and implementation of information and entertainment systems in the car industry, Henfridsson et al. (2014) propose that product designers and developers could make use of complementary architectural frames in order to conceptualise digital and physical design hierarchies as their design and iteration cycles unfold at different speeds; product architectures should not simply be seen as physical hierarchies-of-parts as per the traditional views of product modularisation, but as something that coexists with the network-of-patterns. Whereas the former needs to be fully designed and specified before a product can be transferred to

manufacturing, the latter is a set of abstract patterns which represents the digitally implemented and controlled functionality which remains open-ended and may undergo frequent design iterations. This way, the patterns can be seen as placeholders which can be further developed at a later time. Subsequently, the use of the two complementary frames accommodates the differing levels of material inertia and facilitates asynchronous design cycles during the design and development of products that comprise both the physical and digital elements (Henfridsson et al. 2014).

Moreover, as the affordances and limitations of different components, such as computing capabilities and physical constraints, set the ultimate boundaries on what can be achieved through the digital means, the architecture of a product should be designed in a way which accommodates the changes in and functioning of digital components and computer-controlled functionality. The subsequent section takes a closer look into the intertwined character of design hierarchies in the context of complex product and systems.

2.5 Complex and digitised products

This section takes a look into the definition of robots and autonomous systems and reviews innovation literature in the view of product architectures and organising logic of complex systems and products (Hobday 1998; Prencipe 2000). To this end, the first subsection discusses the purpose and functional principles of robots and autonomous systems in the light of cybernetics, behaviour and the capacity to act. Then, the second subsection looks into the complexity of product architectures and the intertwined design hierarchies of inclusion and control (Murmann & Frenken 2006). In the light of behavioural and technological complexity, robots and autonomous systems can be viewed as complex and digitised products.

2.5.1 Robots and autonomous systems

Robots and autonomous systems can be defined as computer-controlled machines that are designed and built to perform tasks with limited human intervention, with an aim to loosen the coupling between machines and their human operators. Once set in motion, they are expected to produce behaviour

that allows them to operate towards a given goal whilst navigating through the environmental contingencies; they carry out goal-directed plans and actions in response to sensory data they gather from their surroundings (Bekey 2005). However, coupling a machine directly with the surrounding environment does not mean the elimination of human involvement (Mindell 2015). Instead, the type and configuration of human involvement changes (Mindell 2015; Barrett et al. 2012) – while the human operators might be pushed further away from the control of situated action, the role of designers and engineers increases as they produce the means and rules according to which machines operate once they are set in motion. Moreover, leaving machines to their own devices also shifts the locus of (inter)action from the human-machine interface to that between the machine and its broader environment.

Against this backdrop, robots and autonomous systems can be conceptualised as cybernetic systems. Cybernetics is a branch of systems theories (Bertalanffy 1968), which studies the structures, properties and behaviour of control systems through the lenses of communication (information) and feedback mechanisms (Wiener 1965). The central idea of cybernetics is the stability of the system with respect to the given goal (Glanville 1997). The system seeks to approach or maintain its goal by regulating its behaviour in response to environmental contingencies. The paradigmatic example of a cybernetic system is the thermostat that regulates room temperature by adjusting the heating so that the measured temperature (a system state) would correspond to the pre-set target temperature (goal) (Meadows 2009). Ideally, the state of the system should stay, or become, the same as the goal. Whereas the goal and the behaviour of the thermostat tend to be relatively static, the movement of a system or its goal renders the notion of stability to that of dynamic stability (Glanville 1997). For example, the functioning of a self-driving car can be viewed as dynamic stability as the car navigates through the road network and responds to (or absorbs) a variety of contingent events while aiming towards the stable state in terms of a given destination (goal). Moreover, sometimes the goal or target itself may be in the move. Such is the case for example with air defence missiles when they aim at the targets flying in the sky (Glanville 1997).

While the early cybernetics provides concepts and theories that explain control systems and mechanisms, it also brings forward the fundamental limitations of control systems. The law of requisite variety (Ashby 1958) postulates that the control system must be able to absorb and respond to the variety that emerges from the environment, meaning that the mechanism that controls the system should deliver for every input (disturbance) an output (response) that is considered acceptable to avoid a system failure. Furthermore, the second-order cyberneticians make a note that the criteria of acceptability are relative to the observers' preferences (Hayles 1999), highlighting the context-dependence of control systems and regulatory mechanisms. In this light, the context and situation form the mould into which an autonomously operating system must fit in. If the internal structure of a system is not able to capture and operate according to the laws, conventions and meaningful features which characterise its context and environment, it may fail to serve its intended purpose (Simon 1996). Therefore, the apparent complexities of a system's behaviour are largely a reflection of the complexities which emerge from the surrounding environment (Simon 1996). Correspondingly, Maturana and Varela (1992) conceptualise (living) autonomous systems as autopoietic, self-producing and functionally closed, entities, which respond to the environmental inputs according to their internal structure while simultaneously trying to preserve the integrity of their internal organisation. As the cybernetic control systems rely on context-dependent and distributed computation, they are a form of interactive computation (Goldin et al. 2006; Wegner 1997). The notion of interactive computation presents complex and context-dependent computations as long-term relationships, in which the course of computation is contingent upon the information from and interaction with the surrounding environment.

Considering that robots and autonomous systems are expected to perform tasks with limited human intervention, they can also be viewed through the lens of the technological agency as the responsibility of carrying out actions is delegated to machines and computers. In this view, the machines that exhibit some contingent behaviour on their users' behalf are often referred to as agents or complete agents (see e.g. Maes et al. 1999; Shneiderman & Maes 1997; Wooldridge & Jennings 1995; Bryson 2010), yet agency of software or other technological artefacts should not be confused with that possessed by humans

(Kallinikos 2002; Kallinikos 2005). Therefore, the notion of the delegation of agency describes the overall situation in which machines are bestowed with the capacity to act on someone else's behalf, bearing a strong resemblance to the theories that deal with outcome uncertainty, incentives and risk in different principal-agent relationships (Eisenhardt 1989a).

Delegating agency to robots and autonomous systems leads to the reconfiguration of boundary relations among social actors and locales of action and control. For example, Barrett et al. (2012) show how the introduction of a robotic medicine dispenser in a pharmacy setting changes the boundary relations among occupational groups due to the power imbalance over the decisions and priorities which direct the development of the dispenser's behaviour. In addition to organisational power relations, the delegation of agency can be viewed also as a more situated phenomenon. To exemplify, with self-driving cars, control can be dynamically reallocated between the car and the driver throughout the journey (Wray et al. 2016), depending on the road and traffic conditions. Drones used in wars efforts offer another example of the shifting locales of action and control. While drones are able to fly, navigate and track objects as per given targets on their own, a range of choices are still made in remotely located and distant centres where strategic and lethal decisions are made (Gregory 2011). Therefore, the actions and behaviour of robots and autonomous systems emerge from some combination of human action and technological responses to environmental contingencies (Mindell 2015). Moreover, in this light, while machines may possess some situated capacity to act, they to do so within the framework of human purposes and goals – robots and autonomous systems are human creations and would not exist or operate independently of human actions.

To summarise, robots and autonomous systems are defined as cybernetic systems that carry out some particular goal-directed behaviour in a particular context. However, as the behaviour they produce is designed, built, taken into use and controlled by people, they are autonomous only in the weak and limited sense of the word.

2.5.2 Complex digitised products

The behavioural and technological complexity of robots and autonomous systems makes the conceptualisation of the organising logic of such systems challenging. Whereas the literature on modularity holds that modularisation and design rules (Baldwin & Clark 2000) provide a way to distribute and coordinate design and manufacturing activities among different actors (Sanchez & Mahoney 1996), the literature on complex products and systems challenges the applicability of modularisation and modularity (Prencipe 2000). This rests on the observations which revolve around two factors, the lack of mass-market conditions and dominant designs and the intertwined characteristics of design hierarchies in product architectures.

Complex products and systems innovation occurs often in the absence of market conditions and dominant designs of mass-market products (Miller et al. 1995). Complex high-cost systems, such as flight simulators, fighter jets, intelligent buildings, manufacturing systems and nuclear power plants are heavily customised large-scale products which are bought and sold under coordinated and institutionalised market conditions (Miller et al. 1995; Hobday 1998). The specification, development and production may involve suppliers, users, regulators and professional bodies and the end product may lack a dominant design in the sense of mass-market products and industrial process innovation (Utterback & Abernathy 1975; Abernathy & Utterback 1978). The evolution of product architectures and design hierarchies appear to reflect more the progress of underlying technologies (Miller et al. 1995) than the co-evolution of mass-market concepts and corresponding product designs (Clark 1985). For example, from research into the history of flight simulators, Miller et al. (1995) locate the dominant design of flight simulator to its functional principles, to the six degrees of freedom of flight dynamics, and show the long-term stability among suppliers while the underlying technologies that produce the required functionality undergo significant evolution. Since complex products and systems are often relatively unique and project-based, they require a significant amount of architectural knowledge and innovation (Hobday 1998; Henderson & Clark 1990). Therefore, one of the main tasks of a complex system project (or a systems integrator) is to coordinate communication among different

stakeholders and integrate architectural and component knowledge and designs in order to produce specified outcomes (Hobday 1998).

Another observation brings forward the intertwined character of overlapping design hierarchies within product architectures. Autonomous machines comprise sensors and actuators that facilitate the interaction with the surrounding environment. Between the sensors and actuators is located the computational logic and software that transforms the data received from sensors into digital commands that drive actuators. Bekey (2005) defines the software architecture in the context of autonomous systems and advanced machinery as:

"The structure of software, the way in which the robot processes sensory inputs, performs cognitive functions, and provides signals to output actuators" (Bekey 2005, Chapter 5)

Therefore, in the context of robots and autonomous systems, the term software architecture is often used as a synonym for the control architecture, and software architectures typically consist of a spectrum of software that ranges from the low-level hardware control software to the high-level software that performs tasks such as perception, planning, decision-making, reasoning and motion control. Given the variety of software and control architectures in the field of robotics and artificial intelligence (Russell & Norvig 2010), which range from the hierarchical and parallel processing to distributed and multi-agent systems to name a few, we refrain from detailing them here. However, we do make a note that the overall goal of the control software is to transform sensory inputs to action commands and they are of central importance in the view of product and systems architectures.

Murmann and Frenken (2006) highlight the importance of paying attention to both hierarchies of inclusion and control in order to understand the roles and relationships of different parts of some particular artefact.

"We think one additional distinction is absolutely essential to understanding the technological characteristics of an artifact. Complex technological artifacts such as an airplane can be described in terms of two kinds of hierarchies: a hierarchy of inclusion and a hierarchy of control." (Murmann & Frenken 2006, p. 938 cite Wilson 1969)

The hierarchy of inclusion refers to the hierarchical and nested organisation of parts which constitute the physical embodiment of a product. In turn, the hierarchy of control refers to the parts and functional logic which control the operation and behaviour of the embodiment (Murmann & Frenken 2006). To exemplify, the hierarchy of inclusion of an aeroplane consists of sensors, fuselage, wings, engines and landing gear, each of them being composed of their own hierarchy of parts. The hierarchy of control, on the other hand, refers to the mechanisms that control the embodiment; an auto-landing system operates wings flaps, tail fin and engine power based on sensory inputs in order to land the plane safely on the runway (Mindell 2015). This way, the hierarchy of control which produces goal-oriented behaviour operates upon the mechanical and physical substrate that is defined by the structure and features of the inclusionary hierarchy of parts. The intertwined and crosslinked character between the hierarchies of inclusion and control is illustrated in Figure 5.

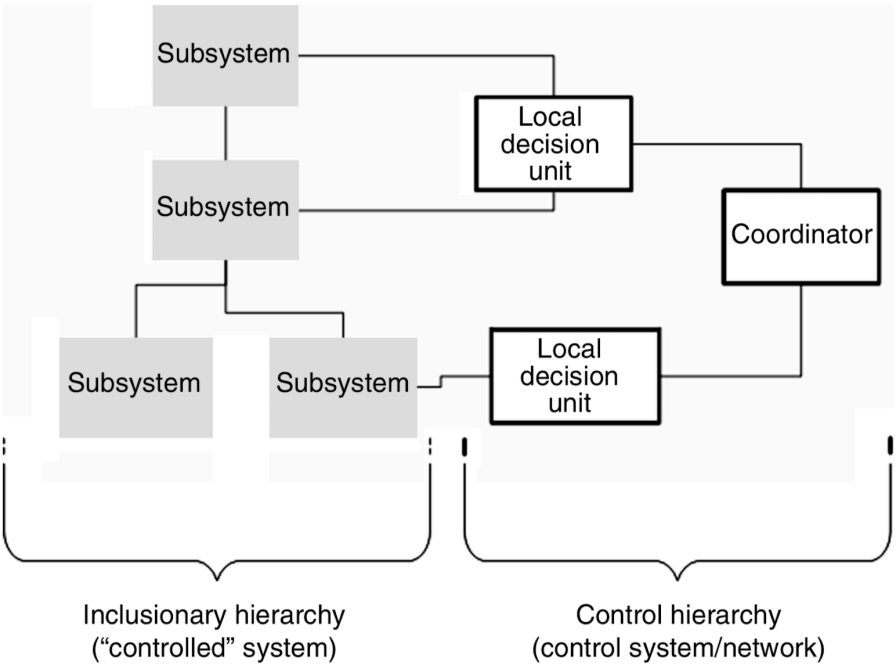

Figure 5: Illustration of the dual-product hierarchy view of complex systems (Lee & Berente 2012)[5]

[5] Republished with permission of Organization Science, from Digital Innovation and the Division of Innovative Labor: Digital Controls in the Automotive Industry, Jaegul Lee and Nicholas Berente, 23/5, 2012; permission conveyed through Copyright Clearance Center, Inc.

Therefore, the hierarchy of control must conform to the underlying dynamics and material properties of the controlled system as well as other environmental contingencies. This dynamic interaction among embodiments, control systems and the surrounding environment can be viewed as a nexus of research interests in the field of robots and autonomous systems (Bekey 2005). Considering that control systems are often implemented using digital means, Lee and Berente (2012) refer to them as digital control systems.

While the hierarchies of inclusion and control describe and analyse the composition of a complex system from two different viewpoints, these two are often deeply intertwined in operational terms. The structural and functional interdependence among inclusionary and control hierarchies often require deep and detailed knowledge of the overall system architecture as well as the functioning of individual components (Prencipe 2000). For example, aircraft engine manufacturers maintain a range of technological capabilities concerning both control system and component technologies in order to maintain their systems integration capabilities (Prencipe 2000). Therefore, organisations which engage in complex innovation that entails multiple technologies often know more than they make as they need to be able to manage and coordinate product development and manufacturing activities (Brusoni et al. 2001). In addition, given the context specificity and variety of functional requirements, dominant designs and design hierarchies are not always readily observable and can occur at different levels of design hierarchies and in different forms, either as stable components or as stable interfaces (Murmann & Frenken 2006). Based on these observations, complex products and systems innovation is postulated to require a breadth and depth of knowledge (Prencipe 2000), a significant amount of mental efforts (Arthur 2009) and occasionally cooperation at an industrial scale (Miller et al. 1995).

Complex products and systems innovation appear to escape straightforward modularisation (Prencipe 2000). The increasing degree of interdependencies among the hierarchies of inclusion and control can reduce the divisibility of innovative labour among product manufacturers and components suppliers even in mature mass-manufacturing industries (Lee & Berente 2012; Henfridsson et al. 2014). While the notion of modularisation rests upon the idea

of separation of concerns by hiding the internal design and implementation of components (Salvador 2007; Baldwin & Clark 2000), Lee and Berente (2012) show that a car manufacturer may take part in component innovation in the development of digital control systems. On a similar note, Henfridsson et al. (2014) show how the intertwined and distributed character of various components of a vehicle information and entertainment system necessitates organisational and technological capabilities that facilitate the integration of various components into complete infotainment solutions. As product manufacturers' role expands, component manufacturers' control over innovation trajectories may become more constrained (Henfridsson et al. 2014).

To conclude, robots and autonomous systems are composed of physical and digital components. Understanding the interdependencies between the hierarchies of inclusions and control are of central concern as the relationship between the two is often highly integral and purpose-specific. Considering the depth and breadth of architectural and component level knowledge required in complex products and systems innovation (Prencipe 2000), modularisation attempts appear elusive in this domain of innovation.

2.6 Problematisation and research question

The review presented above shows that the logic of combination differs among different types of technological innovation. Researchers have developed conceptual frameworks that revolve around modularity, generativity and complexity to conceptualise and highlight particular characteristics of innovation. Much of this literature builds on Simon's (1996) theory of hierarchy and Alexander's (1964) work on the architectural design, which converge on the combination of elements and on the partitioning of a problem space to contain complexity, although they approach the topics from slightly different angles.

Product architectures are often viewed through the lens of modularity (Ulrich 1995; Campagnolo & Camuffo 2009; Salvador 2007). Modularity results from modularisation (Baldwin & Clark 2000), which is a process that decomposes the overall design of a product or product system into modules in a top-down manner so that interactions and interdependencies are greater within the modules than across the modules. Subsequently, interface definitions,

integration protocols and testing standards are specified as design rules, which govern the detailed design and subsequent assembly of modules into complete products (Baldwin & Clark 2000). Modularisation is argued to give rise not only to a product architecture but also to corresponding design and task structures, activities and economic systems that mirror the architecture of a product (Baldwin & Clark 2000).

The applicability of top-down modularisation and isomorphism between product architectures and organisational arrangements have been challenged in the literature on digital innovation (Yoo et al. 2010; Henfridsson et al. 2014) and complex products and systems (Brusoni & Prencipe 2006; Prencipe 2000).

The literature on digital innovation challenges modularisation by presenting a world where products and services do not necessarily result from centralised top-down design and well-crafted design rules (Yoo et al. 2010). Instead, digital innovation is conceptualised to emerge as generative bottom-up combinations of product agnostic components, as assemblages of platforms, applications and services that were not initially designed to be part of some particular product (Zittrain 2008; Yoo et al. 2010; Yoo 2012b). Therefore, the design agency in digital innovation can be described as distributed (Nambisan et al. 2017), transcending the organisational boundaries of knowledge and control (Yoo et al. 2010).

The literature on complex products and systems innovation also challenges the view of modularisation but on different grounds. Considering that the boundaries among product architectures and design and task structures overlap as complex systems incorporate the intertwined and interdependent design hierarchies of inclusion and control (Prencipe 2000; Lee & Berente 2012; Murmann & Frenken 2006), they defy the attempts of modularisation and the subdivision of design and innovative work. Instead, the ability to design and build a well-functioning product requires intimate knowledge of the intertwined design hierarchies at the component and product architecture levels (Prencipe 2000; Brusoni et al. 2001). Considering the level of knowledge and capabilities that are required at different levels, the notion of generativity and the associated

distributedness of knowledge and control seems particularly ill-suited approach in the context of complex products and systems.

Therefore, a contradiction emerges in the view of robots and autonomous systems innovation, as they can be described at the same time as complex and digital. They are complex systems with context-specific requirements and consist of the intertwined hierarchies of inclusion and control. At the same time, control systems are implemented using the methods and techniques of digital computation, which are seen as generative and to transcend well-defined boundaries of knowledge and control.

Problematisation can be phrased as follows: In the context of robots and autonomous systems, would the logic of combination be better understood through the lens of digital or complex innovation? Yet, there is no obvious answer to this question. On one hand, industries such as automotive and aviation acquire and consolidate technologies and knowledge to design and develop proprietary and purpose-specific digital control systems. On the other hand, this approach is unattainable for organisations that cannot afford the consolidation or are not able to commit to some particular product design or technology for periods of time long enough that would warrant the investment. This group includes start-ups, small-medium sized companies, research institutes and universities. These organisations design and develop robot systems by combining technologies and knowledge that are spread across technological domains and communities, bringing the tension between complexity and purpose-specificity of intertwined product architectures and the distributedness of knowledge and skills to the spotlight.

Based on this problematisation, the principal research question is presented as follows:

How can the tension between the specificity of designs and the distributedness of knowledge and control be resolved in the development of complex and digitised products?

2.7 Summary

This chapter reviewed the literature on the organising logic of innovation in the view of product architectures and combination and presented robots and autonomous systems as complex digital innovation. The chapter presented definitions and types of product architectures in the view of the modularity of product systems, the generativity of digital innovation and the specificity of complex innovation. Subsequently, the reviewed literature was summarised to outline problematisation before presenting the principal research question. The subsequent chapter presents the theoretical framework and operative research questions that guide data collection and analysis.

3 Theoretical framework

Building on the problematisation presented in the previous chapter, this chapter presents a theoretical framework that directs the course of this research. This framework provides the foundation upon which tentative a priori concepts are constructed and, in turn, translated into operative research questions, which operate as sensitising devices that inform data collection and analysis. Given the exploratory character of this research, the objective is placed on the identification and description of salient themes and patterns but not to explicate their interactional processes in any detailed manner (Gregor 2006).

The reviewed literature on the organising logics and combination in the view of product architectures builds upon the theories of hierarchy and design (Simon 1996; Simon 1962; Alexander 1964), although different lessons have been drawn from different empirical domains. Much of this research builds on Herbert Simons (Simon 1962; Simon 1996) and Christopher Alexander's (Alexander 1964) theorising on the structural arrangements and characteristics of the category of systems that can be described as complex. Whereas Simon's (1996) work centres on the theory of hierarchy, presenting the architectures of complex systems as nested and recursive hierarchies, Alexander's (1964) main interests are in the synthesis of the form and function and in the partitioning of design problems for containing complexity. While these works differ in certain aspects, they overlap significantly as they both deal with the need and strategies to contain systemic complexity. They show how systemic complexity is reduced through subdivision, by subdividing an overall system into smaller and more manageable constituent elements, subsystems, components and parts. Against this backdrop, complex systems are frequently viewed as nested hierarchies in the innovation and technology management literature (Murmann & Frenken 2006; Arthur 2009).

The following sections describe and develop theoretical framing. First, the early beginnings of systems theoretical thinking are outlined. After that, theories of complex systems are introduced and discussed to provide a foundation for tentative a priori concepts, which are then translated to operative research questions before the concluding remarks.

3.1 Thinking in systems

In general, systems theories are used to study sets of elements in interaction (Bertalanffy 1968). Given the generality of this notion, systems theories are frequently used to describe, explain and occasionally predict a variety of phenomena that revolve around social, natural and artificial circumstances (Merali & Allen 2011). The wide applicability of this general concept has produced a large variety of systems theories and conceptualisations. This variety reflects the variety of empirical phenomena as well as researchers' considerations regarding what kind of systemic theorisation would appropriately capture essential elements, interactions, properties and behavioural dynamics of some empirical phenomenon that is studied as a system. In this view, systems theories and systems thinking provide a wide body of conceptual and abstract knowledge against which different empirical phenomena can be reflected upon. However, regardless of the wide conceptual body of systems theoretic knowledge, research challenges often revolve around the identification and description of an empirical system (Checkland 2000), which is a precondition for mapping it against an analogous conceptual system. Systems theories in their broad variety are widely used in organisation (Boulding 1956; Ackoff 1971), information systems (A. S. Lee 2010; Matook & Brown 2016) and digital innovation research (Hanseth & Lyytinen 2010).

The barebones definition of a system defines a system as a complex of interacting elements (Bertalanffy 1950). While this high-level abstraction tells very little about any particular instantiation of a system or its properties. What differentiates different systems from each other is the number and type of their constituent elements and their respective interactions. These three basic elements are postulated to establish the structure of any instantiation of a system and give rise to its particular properties and behaviour (Bertalanffy 1968). Overall, the systemic view argues for a holistic approach to research.

This is called systems thinking (Checkland 2000), and it seeks to counter the reductionist research approaches which examine and explain phenomena by reducing them gradually into ever smaller entities and relationships. In the reductionist view, the fundamental explanation would eventually be found somewhere at the subatomic level, and the wholes would equal to the sums of

the elements that constitute them. The kind of reductionism that seeks for explanation in terms of ever more fundamental parts (e.g. it's all physics) is referred to as ontological reductionism (Honderich 2005). To challenge the reductionist view, systemic thinking advocates a more holistic approach that appreciates entities as wholes which possess some properties and characteristics that exist only at some given level of analysis. Such properties and characteristics are lost when the wholes are sliced down to smaller and more isolated but more researchable entities because the interconnections and thereby the effects of interactions and interdependencies at some particular level are not considered in the analysis. Consciousness and human life are well-cited examples of the holistic and higher-level systemic behaviours and properties that cannot be fully understood by examining their constitutive elements, such as brain cells, heads, arms or legs, in isolation. Thereby, incorporating the interactions and interdependencies among different elements of a system into data collection and analysis is seen to offer a more holistic view.

Furthermore, although different scientific fields focus on different phenomena, observing different matters, interactional patterns and forces, von Bertalanffy (Bertalanffy 1950) postulates that some of the phenomena studied in different scientific fields and domains may exhibit isomorphic properties when they are examined as conceptual systems and in abstract terms. On these grounds, he advocates the idea of a general systems theory, a general body of knowledge which provides a layer of conceptual abstraction upon which the isomorphism of systemic phenomena across different scientific fields could be studied, and with reference to which abstract methodological tools and theories could be developed (Bertalanffy 1950).

However, when the barebones definition of a system is fitted against some empirical social, natural or artificial phenomena, it becomes soon evident that elements and their respective interactions manifest themselves in multiple patterns, flows, levels, hierarchies and contexts which all can overlap each other in multiple ways. To connect empirical observations with abstract conceptualisations, the phenomenon under investigation needs to be described so that its observed and abstracted features can be considered to form an isomorphic relation to its conceptualisation. In addition, as systems are often

affected by the environment in which they are embedded, it is also essential to include relevant parts of the surrounding environment in the description as well to describe the boundary that is seen to separate a system from its environment. Furthermore, while different system theories offer conceptual and methodological lenses to study and locate a variety of phenomena, the extent to which different phenomena can be systematised is contingent upon their type and origin. For example, simple technologies and closed experimental settings can readily be subjected to systematic analysis, whereas complex and embedded social systems, such as large-scale social and organisational phenomena, may prove too wicked and amorphous for any straightforward analysis or systemisation (Camillus 2008; Rittel & Webber 1973; Buchanan 1992). In this light, the definition of a system and by extension its properties derive from the choices according to which a particular system under investigation is carved out from its natural context and placed into an abstract conceptual framing.

Moreover, ontological holism and reductionism should not be confused with methodological reductionism (Honderich 2005). Indeed, much of the so-called holistic research is reductionist in methodological terms as it seeks to reduce explanations to the smallest feasible number of concepts or variables using the principle known as Occam's razor (Wimsatt 1994). Research projects are often formulated and presented in the form that covers a small collection of concepts that are seen to be related to each other in a way or another so that in conjunction they constitute the phenomenon that is being studied (Gregor 2006).

3.1.1 Complex systems

The intractability of systemic problems increases with the degree of complexity, and the degree of complexity indicates the extent to which a system is amenable to systematic analysis (Bertalanffy 1950). The amenability is contingent on the characteristics of a system, such as the number and type of elements that constitute a system, the extent to which the behaviour of an individual element depends on the behaviour of the other elements of the system, the extent to which a system exchanges energy, matter or information with the environment it is embedded in and its ability to change its behaviour through learning and evolutionary processes (Mitchell 2009). The degree and type of complexity are

contingent on the empirical system under investigation, and a high-level classification of systems according to their complexity is outlined below.

Overall, systems can be divided into different classes along the lines of simplicity, complicatedness and complexity (Mitchell 2009). Systems can be viewed as simple when they are composed of a small number of similar and weakly interdependent elements and are closed from the influence of the outside environment. When the number, type and interdependence among elements increase, and a system becomes more open to external influences, it starts becoming more complicated and/or complex. The difference between complicated and complex can be described along the lines of the linearity and predictability of a system's behaviour (Perrow 1984; Nolte 2015). Designed and engineered complicated systems, such as jet engines and aeroplanes, are expected to operate in a predictable and reliable manner within the bounds of specified conditions and constraints, whereas, for example, the weather system is a paradigmatic example of a complex system due to its unpredictable and emergent character. The notion of emergence holds that the behaviour of a system and its higher-level properties emerge from lower-level interactions as elements interact with each other according to some particular rules without central coordination (Nolte 2015). These unpredictable and emergent characteristics are often defined along the lines of an ability to create a new order, structures or ways of working, or to move a system continually from some state, structure or equilibrium to another, with a more or less predictable manner (Mitchell 2009). Among other criteria, the types of emergence can be also differentiated along the lines of synchronic and diachronic (Bedau & Humphreys 2008). The synchronic refers to some temporal higher-level of systemic phenomena such as human consciousness, whereas the diachronic refers to the events that unfold over longer periods of time, such as history and the evolutionary development of the Internet (Tilson et al. 2010; Zittrain 2008; Hanseth & Lyytinen 2010).

The emergent unfolding of a phenomenon does not necessarily imply that it could not be explained in reductionist terms in principle (Simon 1996; Wimsatt 1972). While it might be impossible to foresee how different development trajectories of a system may unfold and emerge in the future, explanation after

the fact is not totally out of reach if the trails of events can be established. Also, while some systems might, in essence, be just entirely deterministic and predictable, the lack of knowledge of their dynamic properties and initial states may render them emergent and unpredictable for all practical purposes (Simon 1996; Mitchell 2009).

3.1.2 Structures of complexity

When the number of interactions and interdependencies among the elements of a system increases, complexity soon reaches unmanageable levels. This is elaborated by Alexander (1964) and Simon (1996) who also show that systemic complexity can be contained and managed through the process of subdivision and stable subassemblies.

Simon (1996, pp.183-184) defines a complex system as *"one made up of a large number of parts that may have interactions"*. Based on the observations of natural and social phenomena and technological artefacts, Simon (1962; 1996) theorises that complexity reduces when a larger system is subdivided into sets of interacting elements. This is based on the observation that the patterns of interaction and interdependencies among elements are not always uniform. In other words, some elements of a system form clusters (subsystems) within which they are more connected with each other than to the elements which reside outside those clusters. In turn, those subsystems, when combined, form an overall higher-level system and its behaviour. Furthermore, the subsystems themselves are compositions of other subsystems. This recursive hierarchy indicates that higher-level systems are composed of interrelated subsystems, which are themselves composed of some other subsystems until some fundamental level is reached. While the overall complexity of a system can be contained through subdivision, complex systems are not fully decomposable as the overall behaviour of a system is produced through the interaction of interrelated subsystems, thereby rendering complex systems as nearly decomposable.

This nested and recursive hierarchic structure contains interactional complexity and enables gradual development through stable intermediate subsystems (also known as subassemblies that serve as reservoirs of accumulated work and

knowledge). Simon (1996, pp.188-189) demonstrates the stabilising effect of a hierarchic structure in the development of complex systems with a parable of watchmakers. The parable shows how subsystems facilitate gradual and resilient progress towards more complex and higher-level systemic structures. Without stable subassemblies (ready-made subsystems), when a watchmaker is interrupted, the process of assembling a watch would always have to restart from the very beginning. The longer the time that is required to arrive at the completion, the higher is the likelihood of interruption and the subsequent return to the square one. On the other hand, when building upon subsystems, the gradual progression towards a higher-level system is more robust and resilient to interruptions and external disturbances as the process of assembly restarts from the subsystems that have already been completed. This way, stable subsystems are intermediate states that store work and knowledge and building on the previous progress speeds up the development effort.

Alexander (1964) observes that subdividing a design problem into smaller sub-problems is an effective way to contain complexity. Containing complexity is essential when design problems grow in size and complexity, as *"[n]o complex adaptive system will succeed in adapting in a reasonable amount of time unless the adaptation can proceed subsystem by subsystem, each subsystem relatively independent of the others"* (Alexander 1964, p.41).

This is demonstrated with reference to a process of design that is viewed as directed problem solving; the ultimate objective is to create a form that fits into its target context (environment) so that they together form a harmonious ensemble: *"Every design problem begins with an effort to achieve fitness between two entities; the form in question and its context. The form is the solution to the problem. In other words, when we speak of design, the real object of discussion is not the form alone, but the ensemble comprising the form and its context"* (Alexander 1964, p.15). The context refers to the environment in its totality, including natural, social and artificial matters as well as human needs and wants, and it is this contextual totality from which problem descriptions and requirements emerge.

Descriptions and requirements can be considered as design variables based on which the fitness between a form and context is evaluated. Should the interaction among design variables be low, finding a solution to a given problem can be done simply by adjusting variables one by one until the harmony between the form and context is reached. However, when the size and complexity of a design problem grows, that is, when the number of design variables increases, and the variables are heavily dependent on each other so that a change in one variable alters others in a way that is not exactly known, finding a solution becomes increasingly difficult or even impossible. By dividing larger problem spaces into smaller subproblems, spaces from which solutions are searched become smaller and thereby making the challenge of finding an appropriate solution more manageable (Alexander 1964). Design problems become easier to tackle when efforts can be focused on the gradual improvement of subsystems instead of trying to solve the overall problem for all of its interdependent variables at once.

Whereas Alexander (1964) approaches this topic through the lens of architectural design and Simon (1996) in more general terms taking natural and social phenomena and design of technological artefacts into account, they converge on the importance of the subdivision as a method for containing and managing complexity. Subdivision speeds up the search by limiting the problem space solutions are searched from and designed for. The resulting subdivions and subsystems can be seen as essential intermediaries as they provide building blocks that enable the emergence and development of complex and nearly decomposable systems from weakly connected subsystems.

While subdivision into intermediate subsystems shows how the behaviour of a clock can be decomposed into smaller parts (Simon 1996), it also suggests that all parts must be carefully arranged to make a clock work, posing limits to the extent to which a system can be decomposed without losing its particular properties. The near decomposability simply holds that the degree of interaction among the elements within a subsystem is higher than between the subsystems. Therefore, while a system can be decomposed into nearly independent subsystems, a higher-level system behaviour is contingent upon the interaction among subsystems which in combination produce the overall system.

3.1.3 Considerations on the identification of structures

Reducing complexity by subdividing systems into nested hierarchies is by no means straightforward (Alexander 1964). In science, the challenge resides on how to localise and identify relevant subsystems and their respective relationships which are considered to constitute the phenomenon of interest (Kauffman 1970). On the other hand, with the sciences of artificial such as design and engineering, the challenge resides on how to partition a design problem in a way that corresponds to the underlying structure and features of the goals and context of the design problem (Alexander 1964).

Hierarchical structuring is also an essential sense-making device from the human point of view. It allows for us to observe, explain and theorise systemic behaviours in terms of subsystems and their interconnections (Simon 1996) – seeing the world as hierarchies allows for simplified descriptions and explanations as redundancies and superfluous details can be removed. Moreover, Simon (1996) argues that as subsystems interact with each other in an aggregative manner, the details of their internal functioning and interactions can be ignored, and that often relatively little information would be lost by representing complex phenomena as hierarchies. In addition, the definition of the complexity or simplicity of a structure depends on the way and level of description — successful simplification means that *"we must find the right representation"* (ibid p. 215). However, Simon (1996) also points out that if some phenomena do not exhibit hierarchic structure, *"[complex systems] may to a considerable extent escape our observation and understanding"* (ibid p. 207), and thereby transcend our efforts of theorising.

Drawing from the studies of biological organisms, Kaufmann (1970) shows that complex systems can be decomposed and described in multiple ways. As biological systems, such as human bodies, can be seen doing many different things at the same time, there are equally many ways to divide them into subsystems and interactions of interest, depending on what systemic behaviour or feature is under examination and explanation. While attention is focused on certain subsystems and their respective causal relationships, those which are considered irrelevant are ignored. Furthermore, even if multiple studies concern the same system, the subdivision may occur across multiple boundaries at

different levels, yielding non-isomorphic and partially overlapping decompositions (Wimsatt 1972). While this allows for detailed knowledge of some particular aspects of different systems and subsystems, it also produces a multitude of conceptual and theoretical accounts at different levels. Although these accounts may not be readily translatable to each other, Kaufmann (1970) argues that different views are expected to be compatible and non-contradictory as they seek to explain different aspects of the same system. However, reconciling different views may require significant efforts.

The difficulty of localising functionality may explain why *"in-principle reductionist may be at the same time a pragmatic holist"* (Wimsatt 1972, p.67 quotes Simon (1962 p.86)). Wimsatt (1972) explains this difficulty of bringing different views together through the concepts of descriptive and interactional complexity. Descriptive complexity results from the uneven and intertwined spatial distribution of functionality among the different parts of biological and social systems (Kauffman 1970). Different sets of subsystems produce different functionalities, and as some subsystems may take part in the production of multiple functionalities, different theoretical descriptions of the system may become overlap each other in a way that their boundaries do not coincide (Wimsatt 1972). Overlapping theoretical descriptions lead to descriptive complexity, which may require significant work if they are to be reconciled. The interactional complexity, in turn, refers to variety and interdependence of different theoretical descriptions involved in the production of some higher-level systemic behaviour or functionality (Simon 1996). Moreover, Wimsatt (1972) argues that the decomposability of a system should be viewed differently depending on whether we are considering decomposability before or after design or aggregation, and highlights that the hierarchical arrangements that are developed and become more intertwined through evolutionary co-dependent development processes are not readily decomposable whereas some newly engineered systems and subsystems might be. Therefore, a functional organisation that has developed through evolutionary processes does not necessarily correspond neatly to the most readily observable physical organisation. The localisation of functionality and the analysis of a systemic behaviour become increasingly difficult when descriptive and interactional complexity increase (Wimsatt 1972).

The challenge of making sense of complex systems renders requirements specification and engineering equally challenging, in particular when the number of contextual elements and associated requirements is large and they are tightly interdependent (Rittel & Webber 1973; Buchanan 1992). Often, it is easier to point out what appears to be wrong and incorrect than to explain and specify exhaustively why some combination of a form and context would be in the state of complete harmony; incorrectness sends signals, but correctness remains silent (Alexander 1964). However, logically speaking, the absence and presence of misfits are no different. They can be considered as binary conditions for given criteria of fitness, and as long the criteria are listed, it is possible to evaluate for each criterion whether the state of fitness has been achieved or not.

In practice, it is often difficult to specify contexts and criteria down to a level that would leave no room for disagreement or interpretation. This leads to situations where a great deal of information is embedded in task statements and taken for granted assumptions. Based on this reasoning, Alexander forwards that *"the process of achieving good fit between the two entities should be seen as negative process of neutralising incongruities, or irritants, or forces, which cause misfit" (Ibid p. 24).* Therefore, design processes, and therefore by extension engineering, are seen as problem solving which proceeds towards fitness and congruence between a form and its context.

3.2 Operative research questions

The theorisation of the structural arrangements and characteristics of complex systems presented above brings forward the intertwined and multifaceted relationships of separation and combination. Complex natural, social and artificial systems can be viewed as compositions of subsystems, and they are made sense of and more manageable through the processes of decomposition (Simon 1996; Alexander 1964), through the processes of partitioning that seek to subdivide higher-level systems into the lower-level systems and components that in combination compose them. Yet, at the same time, the notion of near decomposability forwards that certain systemic properties and characteristics can only be observed or exist when particular subsystems are brought together and interact with each other. Thereby, the sense-making and operation of

complex systems invariably involve tensions that revolve around separation and combination of different subsystems and elements at different levels.

This tension between separation and combination is manifestly present in the principal research question which brings forward the tension that revolves around the specificity of designs and the distributedness of knowledge and control in the current innovation literature. While this abstract high-level question sets out to explore and make sense of this tension, it is not directly applicable to empirical investigation. As Boulding (1956, p. 197) states, highly abstract notions tend to be almost without content *"… for we always pay for generality by sacrificing content, and all we can say about practically everything is almost nothing"*. Therefore, it is necessary to bring the abstract and high-level theories slightly closer to the empirical domain of research.

Therefore, the abstract theoretical framing is translated to tentative a priori conceptualisation, and the primary research question is translated to the operative questions which can be answered through empirical observation and analysis. The primary research question, as it begins with "*How can…",* hints that there are potentially ways to circumvent and manage such tensions emerging from the separation and combination. This is based on the initial observation that various organisations with limited resources take part in complex digital innovation. However, what is not exactly known is how this occurs and what are the characteristics associated with such affairs.

Robot Operating System (ROS) provides the empirical context in which the dynamics of separation and combination are explored and examined. ROS is a software development framework and an open-source community that supports the development of robots and autonomous systems. It brings together a heterogeneous group of roboticists that share and build upon each other's work. ROS is described in more detail in Chapters 5 and 6, entitled Case description and Results of Analysis respectively.

3.2.1 Subsystems and combinations

Operative questions define the type of information that needs to be collected to answer the principal research question (Hintikka 1999). Based on the

theorisation presented above, ROS is studied through the lens of complex systems and by focusing on different subsystems and their respective combinations (Alexander 1964; Simon 1996).

Given the exploratory character of this research, the effort is first directed to the identification of different subsystems and their characteristics irrespective of their level in systemic hierarchies. Therefore, the first operative research question is formulated as follows:

What are the typical instances and characteristics of subsystems, if any?

The identification of typical instances of subsystems occurs against the conceptual background of nested and recursive structures of technologies as illustrated in Figure 2 in the previous chapter (Arthur 2009; Murmann & Frenken 2006). Being in line with Simon's (1996) theory of hierarchy, it shows how complex technological systems are composed of subsystems, which are in turn composed of some lower level subsystems and so forth until some more fundamental or foundational level of constitutive components is reached.

While Figure 2 presents a four-level nested hierarchy with the system level at the top and the component level at the bottom, it is worth to reiterate that there can be more than four levels hierarchical levels and that technologies can be analysed as technologies at different levels of hierarchy (Arthur 2009). This way, the notion of a subsystem is a frame-dependent concept, and it is used to refer to entities which may reside at different hierarchical levels. Furthermore, the notion of a subsystem is agnostic with respect to the design hierarchies of inclusion and control (see Figure 5 in the previous chapter) (Murmann & Frenken 2006), and it is used to indicate subsystems that can belong either to the hierarchies of inclusion and/or control. Subsystems from the hierarchies of inclusion and control residing at different hierarchical levels and structures are included in the scope of examination.

Following the identification of subsystems, characteristics of subsystems are reflected upon different factors, such as their place among different design hierarchies, operational principles and their role in relation to other subsystems

and higher-level combinations (Arthur 2009; Murmann & Frenken 2006). This is carried out to assign them into distinct categories of subsystems. However, it is worth to note that as these characteristics serve only as sensitising devices (Klein & Myers 1999) and guideposts during data collection and analysis; they are not taken as unconditional criteria of analysis but are expected to evolve and develop as the research unfolds.

Once the instances of subsystems have been established, the focus of research shifts from subsystems on their combinations, with an emphasis to explore how different subsystems combine at and across different levels of design hierarchies. To this end, the second operative research question is as follows:

What are the typical instances and characteristics of combinations, if any?

The purpose of this is to identify and locate instances of combination and examine different characteristics of combination. Different combinations and combinatorial patterns are searched for and subjected for more detailed examination. The guiding signposts are erected along the lines of the reviewed literature, and combinatorial patterns will be reflected in the light of the modularity of product systems (Baldwin & Clark 2000; Salvador 2007), the generativity of digital innovation (Zittrain 2008; Yoo et al. 2010; de Reuver et al. 2017) and the specificity of complex products and systems (Prencipe 2000; Lee & Berente 2012) to ensure that different logics of combination will be considered (Henderson & Clark 1990). Again, these organising logics serve as sensitising devices that guide the researcher to pay attention to particular characteristics of combinations during data collection and analysis, and to reflect the extent to which these notions apply to complex digital innovation.

In more empirical terms, the instances of subsystems and combination in ROS are expected to be found around different digital and physical systems and platforms (de Reuver et al. 2017; Baldwin & Woodard 2008), software and hardware components, system development toolkits and frameworks, standardised interfaces or other boundary resources (Yoo et al. 2010; Eaton et al. 2015). In other words, subsystems are expected be found where previous work has accumulated so as to allow the reuse of existing technologies as well as

the collaboration and interaction among distributed and heterogeneous groups of users and contributors.

The operative research questions and conceptual sensitising devices are derived from the literature and are necessary to make data collection and analysis tractable, yet they are formulated in an open-ended manner to retain interpretive flexibility in the light of emerging evidence. Considering the exploratory character of this work, these tentative conceptualisations (Eisenhardt 1989b) are best understood as initial sensitising devices (Klein & Myers 1999) and are expected to evolve during the course of research (Hintikka 1999). The description and examination of subsystems and their respective combinations within the context of ROS presumable help understand how the tensions between the specificity of designs and distributedness of knowledge and control in complex digital innovation are resolved.

3.3 Summary

This chapter presented the theoretical framing that underpins this research and introduced the operative research questions. The framing builds upon Simon's (Simon 1996) and Alexander's (Alexander 1964) work, which theorises the structure of complex systems as nested and recursive hierarchies, where complexity is contained by subdividing complex systems into subsystems that interact with each other. In this light, tentative a priori conceptualisation and operative research questions build upon the notions of separation and combination in the view of subsystems and their respective combinations. Therefore, research efforts are directed towards the identification and analysis of different subsystems and their respective combinatorial patterns across and at different levels of system hierarchies. The examination of separation and combination are expected to offer insights into how tensions between the specificity of designs and distributedness of knowledge and control are resolved.

4 Research design

This chapter presents the methodological approach and design of this research. This research can be characterised as exploratory and interpretative and is designed as an embedded case study (Yin 2009), which makes use of tentative a priori concepts (Eisenhardt 1989b) as sensitising devices (Klein & Myers 1999) and follows the process of thematic analysis (Boyatzis 1998; Silverman 2015).

Research design, methodological choices and the unfolding of the research process are presented and discussed in the subsequent sections. Section 1 (4.1) discusses the characteristics of case study research to establish its suitability for this research. After that, Section 2 (4.2) outlines the process of thematic analysis, whereas the role and use of tentative concepts are presented in Section 3 (4.3). Section 4 (4.4) that describes the research design, and, subsequently, Section 5 (4.5) presents the rationale and process of case selection to establish the boundaries of research and knowledge claims. Subsequently, data collection and research database construction are described in Section 6 (4.6). The research database contains primarily documentary evidence, such as blog entries, conference presentations, scientific and magazine articles and email archives, which are complemented with the field notes from nonparticipating observation and interviews. Section 7 (4.7) presents the process of data analysis that produced two main outcomes. The first one of them is a case description, which is presented in Chapter 5, whereas the second one, which answers the principal research question using thematic analysis and proposes novel concepts for conceptualising complex digital innovation, is presented in Chapter 6. Before moving on to the case description and other findings, this chapter presents the methodological foundations of this research.

4.1 A case study as an evolving inquiry

Case studies are widely used in organisation and management research (Eisenhardt 1989b). According to Yin (2009), case studies are suitable for answering the *how* and *why* types of research questions under conditions where a researcher holds no control over the unfolding of events, and the phenomenon under investigation is contemporary, occurring in its natural environment. The

outcomes of case studies can be concepts, a conceptual framework or a mid-range theory (Eisenhardt 1989b).

Comparing case studies to experiments, surveys and historical analysis highlights the particular characteristics of case study research (Yin 2009). To exemplify, experiments typically live in artificial habitats and can be constrained, controlled and repeated varying one factor at the time, providing a way for establishing theoretical regularities under well-defined circumstances. While this approach has produced remarkable results in some domains, capturing social and organisational phenomena into laboratory settings remains challenging and ethically questionable (Zimbardo et al. 2000). Surveys are a step closer to the wild, and they can be used to collect information concerning some states of affairs from wider populations. They rely on premeditated sets of questions, often collecting frequencies or respondent's perceptions and reporting, in turn, some statistical descriptions and regularities. However, the emphasis and adherence to premeditated and formalised research designs and protocols tend to introduce rigidity into research processes, making surveys less amenable to projects with exploratory elements and where data is compiled iteratively from different sources and in different forms. More open-ended approaches such as historical analysis and case studies provide research designs that overcome the limitations of experiments and survey-based research designs (Yin 2009) as they accommodate the iterative collection and analysis of multiple types and sources of data. Historical analysis, while being similar to case study research, differs from case studies in that it focuses on the past instead of the contemporary events (Yin 2009). Therefore, the open-endedness of research design, the lack of control from the researcher's part as well as the contextual and contemporary embeddedness of a phenomenon under investigation separate case study research from other modes of social research.

The open-endedness of case study research rests on the analytical choices made by a researcher over the course of research (Bauer et al. 2000). The dependence on researchers' interpretations (Klein & Myers 1999) is prone to subject this type of research to criticism, questioning the generalisability and observer independence of research outcomes.

In the end, ideally, scientific research should be an objective endeavour where outcomes do not depend on the interpretations made by some particular researcher. The positivist research paradigms seek to obtain objectivity through the premeditated research designs and mechanistic methods. However, choices made during research design, such as the formulation of a research question, questionnaires and methods of analysis as well as the decisions on whom to ask from among others, bring interpretative elements into positivist research, even if this is not always fully acknowledged (Bauer et al. 2000). In turn, in so-called interpretative modes of research, objectivity cannot be claimed by hiding behind the facade of methodological formalism. The open-ended research design exposes researchers' role and decisions in data collection and analysis.

The open-endedness, however, does not mean that a researcher should do away with methodological theories and tools altogether. Instead, researchers should use methodological tools to build procedural rigour into research activities and to reflect on which particular research methods and avenues should or could be pursued or discarded (Eisenhardt et al. 2016). Methodological rigour along with necessary justifications allows a reader to examine the reasoning behind and credibility of research outcomes. However, although methodological handrails can help a researcher over the narrow stretches, they provide very little in the way of definitive guidance regarding the paths a researcher should take or what to make of them.

The relationships between a researcher's choices and interpretations can be viewed in the light of definitory and strategic rules. Like games, scientific inquiry can be considered in terms of definitory and strategic rules (Hintikka 1999). Definitory rules define a game and describe the moves which are possible and admissible. However, while definitory rules define a set of possible moves, they are not informative regarding the utility of any particular move, as the utility of a move depends on the overall situation where it is taken. The rules which take utility into account are called strategic rules. While strategic rules have to conform to definitory rules, as otherwise such moves would not be permitted, their utility derives from the goals and environment in which they are taken and the overall context of strategies they are members of. In this light, if the overall context of an inquiry is not fully known in advance, the successful

formulation of most profitable sequences of moves remains a precarious effort. Therefore, the upfront research design of an exploratory research is expected to change as the knowledge of the overall context is limited at the beginning and increases as the research project unfolds.

To formalise this logic of scientific inquiry, Hintikka (1999) develops what he calls the method of interrogative inquiry. While the formal logical presentation of the method is not of interest here, the formalisation however neatly presents the iterative and interrogative unfolding of exploratory research. Interrogative inquiry entails two participants, an inquirer and nature. The inquirer is looking for an explanation that answers the principal question. At any given moment, the inquirer may decide to either proceed a step further using logical deduction on the basis of existing facts or, alternatively, present an operative question to nature[6] in the hope of obtaining some new facts. With the newly acquired facts (if nature answers), the inquirer may again proceed further either by performing a new deductive step on the basis of the present facts to refine the working hypotheses or pose another question in the hope of obtaining new facts that are able to confirm or reject the working hypothesis based on which the operative question was constructed. The interplay of logical deduction and interrogation continues until all necessary facts are in place so that the initial principal question can be answered.

Hintikka (1999) separates purely deductive and interrogative reasoning respectively as trivial and non-trivial, associating them with Charles S. Peirce's corollarial and theorematic reasoning respectively, forwarding that:

"[T]heorematic inferences are the ones which introduce a new individual (variables) into the argument, whereas corollarial merely traffic in the individuals which have already been considered in their premises" (Hintikka 1999, pp.7-8).

Non-trivial inference implies the imagination of possible worlds and subsequent systematic probing of the validity of those worlds (Weick 1989). This may open

[6] Nature is broadly defined. It can be either nature as natural nature, or it can be a computerised database, or any of many other things that may provide an answer, and they may differ wildly in their structure and information they hold.

up new lines of argumentation by bringing in new evidence, which in turn may trigger a change in beliefs and working hypotheses, although not necessarily doing so if questions are poorly formulated or nature does not answer. Moreover, Hintikka (1999) notes that deduction and interrogation are the two sides of the same coin; the difference should be considered rather as a matter degree than a difference in kind, as a sort of sliding scale between the two opposite ends of the spectrum where the position of a pointer depends on the number of new variables being introduced into inquiry.

Although Hintikka (1999) formalises the logic of interrogative inquiry, outlining its definitory rules, we are none the wiser when it comes to defining the strategic utility of any particular move. What are the presuppositions to uphold and in which way what is known should be transformed into new sequences of questions and answers? How could a researcher establish the validity of any chosen course of events and outcomes in a convincing manner, as it remains that the formulation of strategic rules resists formalisation because the efficiency of moves depends on the environment (e.g. what can be learned from data) and ultimate goal (e.g. the aimed contribution) in which the moves are taken, both of which are not fully known when an open-ended research project begins.

The iterative and interrogative process builds upon reasoning, questioning and interpretation and requires readiness to adjust for emerging avenues of research throughout the course of research. This corresponds closely to the underlying assumptions and open-endedness of interpretative case study research. The in-depth investigation based on rich data and readiness to accommodate a variety of data from different sources render case study design as an appropriate approach for the research projects that seek to explore and develop a deeper understanding of the phenomenon under investigation, building upon prior and newly acquired facts in an opportunistic but goal-directed manner.

Whereas the exploratory approach can be described as disciplined imagination (Weick 1989), the process and evidence that give rise to the final conclusions should be made explicit to a reader to allow her to evaluate the reliability of

conclusions. The subsequent section discusses the criteria of reliability in more detail.

4.1.1 Quality criteria

The quality of research can be evaluated in terms of construct validity, internal validity, external validity and reliability (Gibbert et al. 2008; Yin 2009), each of them evaluating a particular aspect of research design. A research project should be designed and executed in a manner that reaches high quality in all four above-mentioned areas. Therefore, before presenting the design of this research project, it is beneficial to revisit the quality criteria to set the goalposts against which the design can be evaluated.

Construct validity concerns with data collection and deals with the operationalisation of the concepts that guide data collection. To minimise subjective and observational biases, a researcher should provide clear conceptual definitions and describe the features based on which the concepts can be located in the empirical evidence (Yin 2009). Construct validity can be increased using empirical triangulation, that is, using multiple sources of evidence as well as establishing a coherent "*chain of evidence"* (Yin 2009, p.42), which details the path from research questions to conclusions, allowing a reader to reconstruct the collection and processing of data (Gibbert et al. 2008).

Internal validity can be understood as "logical" validity, meaning that a researcher is expected to demonstrate sound and logical treatment of data during analysis (Yin 2009), crafting a plausible line of argument which is sufficiently compelling to warrant the conclusions of research (Gibbert et al. 2008). However, facts that constitute evidence and their respective relations may not always be readily observable, necessitating interpretation and inference from researchers' part. If the researcher fails to include some crucial facts in the analysis, the resulting findings may rest on spurious evidence and treatment. Depending on the type of evidence and goals of research, a variety of analytic strategies and techniques can be employed to mitigate the risks concerning the internal validity, such as systematic explanation building and evaluation of explanations against predicted patterns and other theoretical propositions (Yin 2009; Gibbert et al. 2008). In addition, evidence should be presented separately

from its interpretation in order to expose the link between the data and its interpretation as it facilitates the examination of alternative interpretations.

External validity indicates to what extent final conclusions are expected to be generalisable or applicable outside the empirical research setting (Yin 2009; Gibbert et al. 2008). Broadly speaking, there are two main strategies towards generalisability. Statistical (positivist) research methods pursue this goal through representative sampling and the subsequent generalisation to corresponding populations. In contrast, interpretive studies with a single or small number of cases seek to construct analytical generalisations with an aim *"to generalise a particular set of results to some broader theory"* (Yin 2009, p.43), which can then be generalised across analogous settings. The generalisation across analogous organisational settings is often a primary goal of information systems research (Seddon & Scheepers 2015).

There are alternative viewpoints to generalisation, such as the generalisation from data to descriptions or from descriptions to theory (A. S. Lee & Baskerville 2003). Seddon and Scheepers (2015; 2012) take a processual view on generalisation, presenting it as a logical argument that spans across different phases and settings of research.

"A research generalization is the researcher's act of arguing, by induction, that there is a reasonable expectation that a knowledge claim already believed to be true in one or more settings is also true in other clearly defined settings." (Seddon & Scheepers 2015, p.38; Seddon & Scheepers 2012)

As parsimony and generalisability are hallmarks of theoretical contributions (Eisenhardt & Graebner 2007), research projects can be viewed as a series of abstractions. Step by step, selected features present in data are preserved while others are either discarded or established as qualifying conditions, moving gradually from data towards theoretical generalisations, which can be transferred across different settings according to some logical of comparability.

Finally, the overall reliability of research refers to the absence of errors and minimisation of biases (Gibbert et al. 2008; Yin 2009). In principle, should another researcher wish to carry out the same research again, she should arrive at the same results. For this to be possible, the research process and outcomes

must be described clearly and transparently. To this end, the researcher is advised to develop a case study database (Yin 2009) and document data collection and analysis to establish an audit trail that allows the replication of research. To ensure quality and generalisability of research outcomes, a researcher should make use of techniques, which increase construct, internal and external validity and lead to overall reliability and transparency. The subsequent section presents the analytical method adopted in this research.

4.2 Thematic analysis as iterative abstraction of patterns

Thematic analysis is a widely used research approach in qualitative social and organisation research. Regardless of the objectives of research, the search for patterns, themes, concepts and categories and their relationships is a common research procedure (Boyatzis 1998; Bryman 2015). This search plays an important role for example in the grounded generation of theory, narrative analysis and qualitative content analysis, and different theoretical and methodological traditions have developed a variety of methods, techniques and heuristics of search that serve particular analytic and theoretical objectives (Bryman 2015).

A theme is a central construct in thematic analysis and thereby requires a further elaboration, especially considering that as an abstract concept it is prone to attract multiple interpretations. Bryman (2015) defines a theme as a *"category identified by the analyst through his/her data"* (ibid p. 584). Moreover, themes are expected to be related to the focus of research, derive from the patterns identified in the transcripts and field notes, and to *"provide the researcher with the basis for a theoretical understanding of his or her data that can make a theoretical contribution to the literature relating to the research focus"* (ibid p. 584).

In this light, the thematic analysis appears similar to grounded theory (Glaser & Strauss 1967). While thematic analysis and grounded theory resemble each other, they differ in their aims. The method of grounded theory is concerned with theory generation through iterative and parallel data collection and analysis until theoretical saturation is reached (Glaser & Strauss 1967; Corbin & Strauss 2008). During this process, the method of grounded theory seeks to

extract concepts from raw data by coding and labelling them into conceptual categories first through the process of open coding, and subsequently by building linkages among categories through the processes of axial coding (Corbin & Strauss 2008). On the other hand, while thematic analysis also aims at uncovering thematic patterns and linkages (Bryman 2015), it does not necessarily aim at developing a new theory (Urquhart 2013). However, this distinction hinges upon where a boundary between a new theory and any theoretical contribution is drawn. According to Eisenhardt (1989b), a range of theorising efforts, such as conceptual development, theoretical propositions and mid-range theories can be viewed as theoretical contributions.

Thematic analysis is not intrinsically linked to any particular theoretical or conceptual framework. While different frameworks, such as narrative analysis or critical discourse analysis, often carry a number of assumptions concerning the nature of data and what particular features of data represent (Bryman 2015; Braun & Clarke 2006), the method of thematic analysis is tasked simply with finding repeated patterns of meaning thereby separating the underlying theoretical assumptions from the process of analysis. The meaningfulness of identified patterns is contingent upon the theoretical framing and objectives of a research project (Braun & Clarke 2006). Thematic analysis is a processual framework for data analysis, which describes the main phases of data analysis process at a level that is abstract and generic (Bryman 2015). The aim of explicating the process steps is to make the process more transparent and replicable and highlight the importance of following coherent and consistent approach during data analysis (Braun & Clarke 2006).

The process of thematic analysis unfolds as follows (Bryman 2015). To begin, the researcher makes herself familiar with the contents of the research database to establish a general overview and understanding of the data being analysed. After this, the process of initial and open coding begins, and this may lead to a large number of scattered and incoherent codes and categories. Then, the researcher iterates and rearranges the initial codes into higher-level categories, and this is followed by a round of iteration during which the higher-level categories are further examined and arranged into themes. Throughout data analysis, a researcher can review and adjust categorisation depending on the

evidence and desired level of abstraction. Once a set of themes has been established, they are described and explained. As a final step, the linkages among different themes can be examined and conceptualised before the production of a final report that describes the resulting themes and their interrelations. The outcome can be presented in the form of a summary table that outlines and describes the prominent themes and categories with examples and descriptions, which illustrate their salient characteristics.

As the purpose of the thematic analysis is to identify and report themes in data (Braun & Clarke 2006; Bryman 2015), what counts as a theme needs to established. According to Bryman (2015), one of the most common criteria that warrants a theme is repetition. The recurrence of a particular pattern may occur within the boundaries of one type of evidence or alternately across various types and sources evidence. However, not all repetitions are equally important with respect to research objectives. Some patterns may frequently repeat themselves in data, yet if they are not relevant in the light of research objectives, they should not be included in the analysis (Bryman 2015). Other common heuristics revolve around the identification of similarities and differences among different patterns. Moreover, themes can be identified at the level of manifestation or interpreted as latent and underlying themes (Boyatzis 1998). At the level of manifestation, patterns are readily observable, whereas latent patterns can be considered as hidden and underlying causes of phenomena.

The observation of manifest patterns tends to offer descriptive accounts, whereas the identification of latent themes require interpretation from researchers' part (Boyatzis 1998). As the exposition of latent patterns cannot rely solely on description, it necessarily includes elements of conceptual and processual sense-making and theorising (Braun & Clarke 2006).

Furthermore, the coding process can be either data-driven or theory-driven (Urquhart 2013; Boyatzis 1998). In the case of data-driven analysis, a researcher approaches data without analytic framework and discovers and derives categories and themes inductively from data. In turn, a theory-driven approach adopts existing theories as a starting point and analyses and evaluates data with reference to a theoretically-driven conceptual framework. In practice, research

projects often fall somewhere between the two and combine both data-driven and theory-driven approaches.

To conclude, thematic analysis is an iterative process for identifying categories and themes in empirical evidence. Categories and themes can be viewed as recurring patterns that represent a concept or idea that is relevant in the light of research objectives.

4.3 Role of tentative a priori concepts

The development of tentative conceptualisation and operative research questions is described in Chapter 3, and their methodological role in this work is elaborated in this section. This is necessary in order to establish the boundary between theory-driven and more exploratory data-driven phases of this research. As discussed in the previous section, thematic analysis can be theory-driven or data-driven (Urquhart 2013; Boyatzis 1998), and that research projects often combine these two approaches. This is also the case here. The process of analysis begins as theory-driven, but as the analysis proceeds further, the approach shifts from the theory-driven to data-driven as patterns are abstracted from data in order to developed novel high-level categories and themes. The role and function of a conceptual framing are discussed below.

The role and function of conceptual framing vary depending on the type and purpose of research. In general, a conceptual framework tends to correspond tightly to its theoretical premises in research projects that test theories, as this allows for unambiguous testing and reporting of the validity of a theory (Yin 2009). To contrast, in the context of more open-ended and exploratory research projects which may target at theory generation, conceptual framing is better understood as a sensitising device (Klein & Myers 1999); it directs attention during the data collection and analysis but is not guaranteed to secure its place in the final results, concepts or conclusions (Eisenhardt 1989b). Tentative a priori conceptualisation serves as a scaffolding which can be discarded when it is no longer needed.

In this research, the conceptual framing and tentative a priori concepts are used as sensitising devices (Klein & Myers 1999; Eisenhardt 1989b); they direct

attention to the matters which are considered relevant with respect to the objectives of this work. While the existing literature and theories provide a starting point and guide through the course of research, they are not held onto rigidly and not taken as an ultimate arbiter on what the researcher should see or report as this may lead to a suppression of conflicting evidence and potentially revealing observations (Walsham 1995). Instead, the aim here is to remain sensitive to any emerging themes and be prepared to shift the focus of research if the evidence at hand so warrants. While the readiness to modify initial assumptions in the light of new evidence may shift the focus, it also enables the introduction of new concepts, constructs or theories (Eisenhardt 1989b).

To summarise, while theoretical framing and conceptualisation help focus attention to the matters and aspects that are seen relevant in the view of the research objectives, in the context of this work such framing is better understood as a flexible sensitising device (Klein & Myers 1999) instead of a rigid frame that must be adhered to at all cost.

4.4 Research design framework

This section introduces the research design framework followed in this research. According to Yin (2009), every empirical research project has a plan which resides somewhere between the implicit and explicit ends of a spectrum. In the light of construct validity, internal validity, external validity and overall reliability and transparency, research design should be articulated in a way that is closer to the explicit than the implicit end.

To this end, Eisenhardt (1989b) offers a stepwise framework for designing and conducting case study research. The framework combines elements from the case study design, qualitative methods and grounded theory and presents a series of steps which describe main research activities along with their respective rationale. The framework is presented in Table 1, and it serves as a template with reference to which this research is designed. The steps listed in Table 1 are described in the subsequent sections to present the methods and procedures that lead to the final conclusions of this work.

Nr.	Step	Activity	Reason
1	Getting started	Definition of research question Possibly a priori constructs Neither theory nor hypotheses	Focuses efforts Provides better grounding of construct measures Retains theoretical flexibility
2	Selecting cases	Specified population Theoretical, not random, sampling	Constrains extraneous variation and sharpens external validity Focuses efforts on theoretically useful cases-i.e., those that replicate or extend theory by filling conceptual categories
3	Crafting instruments and protocols	Multiple data collection methods Qualitative and quantitative data combined	Strengthens grounding of theory by triangulation of evidence Synergistic view of evidence
4	Entering the field	Overlap data collection and analysis, including field notes Flexible and opportunistic data collection methods	Speeds analyses and reveals helpful adjustments to data collection Allows investigators to take advantage of emergent themes and unique case features
5	Analysing data	Within-case analysis	Gains familiarity with data and preliminary theory generation
6	Shaping hypotheses	Iterative tabulation of evidence for each construct Replication, not sampling, logic across cases Search evidence for "why" behind relationships	Sharpens construct definition, validity, and measurability Confirms, extends, and sharpens theory Builds internal validity
7	Enfolding literature	Comparison with conflicting literature Comparison with similar literature	Builds internal validity, raises theoretical level, and sharpens construct definitions Sharpens generalisability, improves construct definition, and raises theoretical level
8	Reaching closure	Theoretical saturation when possible	Ends process when marginal improvement becomes small

Table 1: Research design framework after Eisenhardt (1989b)

In this framework, Step 1 focuses on the formulation of research questions and tentative a priori concepts that guide the process of research. To this end, a principal research question was derived from the reviewed literature in Chapter 3, and a priori concepts and operative research questions were developed in Chapter 3. To answer these questions, this research is designed as a case study that follows the process of thematic analysis. The underlying principles of case studies and thematic analysis were presented above in Sections 1 and 2 (4.1, 4.2), and the role and use of tentative concepts were discussed on Section 3 (4.3).

After having established the foundations and research questions, Step 2 deals with case selection. The selection of ROS as an embedded case study is presented in Section 5 (4.5). While ROS can be considered as an extreme and revelatory case study in the light of previous research into digital innovation, it can also be viewed as a typical example of complex digital innovation in the field of robots and autonomous systems. Steps 3 and 4 concern with crafting instruments and protocols and entering the field. To this end, the process of data collection and the construction of a research database are described in Section 6 (4.6). The research database contains primarily publicly available documentary evidence, which is complemented with the field notes from nonparticipant observation and interviews. The contents of the database are detailed in appendices. Steps 5 and 6 focus on the processes of analysing data and shaping of the findings into potentially new concepts, constructs or theories Section 7 (4.7) describes how the content of the research database is processed thematically to analyse subsystems, combinations and their salient characteristics with an aim to establish a rich and detailed understanding that leads to novel conceptual propositions. Steps 7 and 8, the enfolding literature and reaching closure, are presented in Chapter 7 where conceptual findings are discussed in the light of the literature reviewed in Chapter 2.

4.5 ROS as an embedded case study

Case selection is one of the most important phases in case study research. As case studies build upon the detailed examination of a single or small number of cases, case selection defines what can be learnt and to what extent learnings can be generalised and transferred over to different but similar settings. Ideally, the

selected cases(s) should be informative, realistic and be able to offer novel insights (Yin 2009). This section outlines the selection and definition of ROS as an embedded case study.

Considering that the selection of a case sets boundaries to the extent of knowledge claims, in Flyvbjerg's (2013) terms, carrying out a case study is more of a choice of the units of study and the definition of its boundaries than adhering to any particular method of data collection or analysis:

"The decisive factor in defining a study as a case study is the choice of the individual unit of study and the setting of its boundaries, its 'casing' to use Charles Ragin's (1992, p. 217) felicitous term." (Flyvbjerg 2013, p.169)

In this view, casing means carving a particular case out of some wider context, potentially along with its embedded units of analysis (Flyvbjerg 2013). Wider context represents the environment in which a case is embedded, and it provides a backdrop against which some particular case study can be considered as a separable unit of analysis. Furthermore, case studies can be designed to draw their lessons from a single case or multiple cases, which can be either holistic or embedded. The word holistic indicates the equivalence between a case and unit of analysis, whereas the word embedded signals the presence of more than one units of analysis within a particular case (Yin 2009). The use of embedded units of analysis is beneficial in situations where a more focused and detailed view of some particular aspects of the case is required. However, when relying on multiple embedded units of analysis, the units of analysis and respective unitary findings must be brought together to establish final conclusions at the level of an overall case (Yin 2009). In this view, embedded case studies can be defined as hierarchical structures that consist of three nested levels: the wider context that provides the environment within which the case resides, the case itself and the units of analysis that are embedded in the case. While there is no right way to draw boundaries between the context, case and units of analysis, the way these boundaries are drawn must be explicated as it defines the boundaries of knowledge claims (Flyvbjerg 2013).

Moreover, research projects are typically designed to serve to particular theoretical objectives. Depending on the objectives, case studies can be viewed

as critical, extreme, typical, revelatory or longitudinal (Yin 2009). Critical cases are used to establish validity of theoretical propositions, while extreme cases report about matters which can be considered somewhat deviant or unusual, and potentially calling for novel conceptualisation and theorising in order to explain observed variations (Flyvbjerg 2013). In turn, typical cases are seen as representative and paradigmatic, informing about some common and prevalent state of affairs. Revelatory cases, on the other hand, observe and analyse situations that have been previously inaccessible to researchers, thereby providing opportunities for novel insights and theorising. Longitudinal case studies focus on phenomena which unfold over longer periods of time. As this categorising is relative to the goals of a research project, prior knowledge, and research environment, any particular research design may belong simultaneously to more than one category (Flyvbjerg 2013).

4.5.1 The Robot Operating System

The Robot Operating System (ROS) (Quigley et al. 2009) was selected as a case to study in this research. ROS is a software development framework, and the wider context of ROS can be described as software development for robots and autonomous systems. During the pilot study phase, several case candidates were considered. The candidates ranged from individual robot development projects to different proprietary and open-source software development frameworks. In the end, ROS was considered to provide a solid foundation in the view of the generalisability of research results. The reasoning behind the case selection is outlined below.

Individual robot development projects were considered and discarded first. While the history of information systems research has shown that much can be learnt from detailed examination of individual projects in different socio-technical settings, it was however thought that concentrating on an individual project might steer the findings towards some project-specific aspects, posing a risk to generalisability. After that, proprietary and commercial software development frameworks were considered and discarded. This was done on the basis that they are often tightly-coupled to some specific-purpose hardware or domain of application, thereby potentially steering findings towards some domain-specific aspects. After deciding against individual projects and

commercial applications, the focus shifted on widely used open-source frameworks. From open-source frameworks, the Orocos project (Bruyninckx 2001) was considered first. While it is being actively developed and used, has a long history and is well received in the robotics community, it was set aside as it is geared more towards industrial applications and hard real-time control systems. Also, as some of its core components have been integrated with ROS and it can be used alongside ROS, it was concluded that ROS could provide a more generalisable view on complex digital innovation. In addition, as other reviewed open-source frameworks (Kramer & Scheutz 2006; Iñigo-Blasco et al. 2012) appeared to be less vibrant or geared towards some specific purposes and applications, they were not considered further.

During the pilot study phase, it become apparent that ROS has attracted much attention over the past ten years. It is a vibrant community that attracts roboticists globally and across different application domains. The source code is publicly available and licensed under the conditions which allow it to be used, modified and distributed freely for research purposes and in commercial applications. Although the development of the core functionalities and elements of ROS is driven and coordinated by the Open Source Robotics Foundation (OSRF), design and development efforts are not confined inside the boundaries of a single organisation. Instead, ROS brings together a large and increasing number of users and contributors from academia and industry as well as corporate and government funding. Therefore, it is not surprising that ROS emerged frequently in discussion during the pilot study phase. It was debated in various robotics events and many people appeared to have an opinion of it, either positive or negative. This warranted interest and further examination.

ROS is seen to provide a good foundation for generalisable findings. Although robot software development frameworks are still looking for their shape, ROS offers a relatively representative view of the current state of affairs. It is gaining increasing traction and is also used in commercial applications, although currently it is primarily used for research and development purposes. Moreover, although other software development frameworks are also available, ROS is one of the more popular ones, and it is occasionally referred to as a de facto standard in the robot software development (Sterling 2013). Since it is widely

used, it offers a broad view that enables the examination and evaluation of common problems and solutions as they manifest themselves at the level of a software development framework that caters a broad community. As the ROS website states, ROS supports collaborative software development for robots and autonomous systems, providing technological tools, capabilities and a vibrant community that bring together roboticists and software developers globally. The wide adoption of ROS is seen to provide a foundation for the generalisability across different organisational settings since the organisations that use ROS and, more broadly, develop control software for robots and autonomous systems, are subjected to similar organising logics of combination in the view of product architectures.

Moreover, as ROS provides a framework for the robot software developers engaged in complex digital innovation, drawing lessons from it and reflecting them in the light of the current theories of digital innovation presumably provides a fruitful opportunity to contribute to the literature on digital innovation. Although ROS can be considered as a typical case of digital innovation in the field robotics, especially in the context of research and new product development (Fichman et al. 2014), it can also be seen as a revelatory case in the view of digital innovation research, which has so far focused on the processes of digitalisation in the contexts of digital infrastructures (Tilson et al. 2010), platforms (de Reuver et al. 2017), mobile devices (Eaton et al. 2015) and digitised products (Yoo et al. 2010; Henfridsson et al. 2014).

4.5.2 Embedded units of analysis

Following the design principles of embedded case studies, embedded units of analysis are used to structure the process of data collection and analysis (Yin 2009). In this research, the embedded units of analysis derive from the tentative a priori concepts and operative research questions which were developed in Chapter 3.

With reference to the first operative research question, the first embedded unit of analysis focuses on *subsystems* (Simon 1996) at different levels of design hierarchies of inclusion and control (Arthur 2009; Murmann & Frenken 2006). However, it is worth to note that the concept of a subsystem as a unit analysis

remains somewhat generic and abstract; while it is not a carefully detailed and specified instance or pattern, it is expected to direct attention towards potential instances of interest (Klein & Myers 1999), although much of this process rests on the researcher's interpretation. Subsystems can manifest themselves in multiple forms, places and hierarchical levels and serve a variety of functions (Murmann & Frenken 2006). For example, in terms of digital computation, software libraries, modules, packages, databases, platforms, software development toolkits, frameworks as well as computing hardware could be viewed as instances of subsystems. Similarly, in terms of hardware, different components, embodiments, parts, machines and motors can be viewed as instances of subsystems. Many more examples can be easily found in different areas of socio-technical systems and social organisations (Simon 1996).

In this light, the concept of subsystem as an embedded but generic unit of analysis can be viewed as a dragnet that trawls through data in the search of objects and artefacts that may prove informative. Yet, it remains as the researcher's task to discriminate which ones of them warrant further analysis and examination. While this loose formulation directs focus on specific matters during data collection and analysis, it also retains a high degree of interpretive flexibility that is necessary in exploratory research.

With a reference to the second operative research question, the second embedded unit of analysis centres on *combination* (Arthur 2009; Murmann & Frenken 2006), that is, how different subsystems (Simon 1996) are bought together to create robot systems that produce autonomous behaviour. Again, the concept of combination remains abstract serving the purpose of directing attention to the matters that deal with combinations, that is, how different subsystems related to each other and are joined together – yet it leaves it to the researcher to decide and explain why some particular combination and its respective characteristics are included in the analysis. To provide an example, in the context of software engineering and digital innovation, boundary resources (Eaton et al. 2015) in their variety of forms such as interfaces, standards and documentation can be seen as instances and manifestations of combination.

The characteristic of combination can be further reflected in the light of the patterns of combination that were discussed in the literature review. For example, with the lens of modularisation the extent to which complex digital innovation can be subjected to centralised top-down design to produce stable core elements and well-defined interfaces (Baldwin & Clark 2000; Salvador 2007) can be evaluated. In turn, the lens of generativity (Zittrain 2008) can be used to probe the extent to which complex digital innovation can be subjected to distributed design agency, knowledge and control, and to what extent end-product agnostic subsystems are amenable to generative combinations. The layered modular architecture provides a lens (Yoo et al. 2010) for evaluating the dynamic interplay of physical and digital components within the context of digitised products that combine the modular and generative product hierarchies. Furthermore, the notion of specificity (Prencipe 2000; Lee & Berente 2012) probes the extent to which generative combinations can be said to produce highly integral and purpose-specific behaviour. In addition, the notion of architectural innovation (Henderson & Clark 1990) can be evoked to provide a lens against which the reconfigurations of relationships among the core concepts and components can be identified.

To conclude, the boundaries of this embedded case study are drawn as follows. The wider context is the software development for robots and autonomous systems and ROS serves as a case study that is carved out of this context. ROS can be seen as an extreme or revelatory case in the view of digital innovation research, even if it could be viewed as a typical and representative in the domain of robotics. Within ROS, the collection and analysis of data centres around two embedded units of analysis, subsystems and combinations. They are used to identify and locate the empirical objects that could warrant further examination and provide themes and patterns to answer first the operative and then finally the principal research questions. While these embedded units of analysis help get data collection and analysis started, they remain tentative and subject to interpretation throughout this research. The next section describes the process of data collection.

4.6 Data collection

Data collection was carried out to serve two different but related purposes. The first of them is to develop a case description in order to become familiar with ROS and its evolution over time. This provides a well-grounded view on ROS and the surrounding community, thereby providing a foundation for more detailed analysis. The second is to compile a rich body of evidence that enables an in-depth investigation into the organising logic of complex digital innovation. The collected data was stored into a research database, and the collection and development of the contents of the database are described and discussed below.

To explore and answer the *how* and *why* types of research questions typical to case study research, a well-rounded and comprehensive view of the phenomenon is needed (Eisenhardt 1989b; Flyvbjerg 2013). According to Eisenhardt's (1989b) research design framework, the use of multiple data collection methods and triangulation strengthen the grounding of theoretical findings. In addition, carrying out data collection and analysis in parallel provides an opportunity to adjust data collection processes and strategies if and when needed. Therefore, while the initial research design provides a necessary starting point, the application of tools and protocols remains flexible and open for new avenues of research to emerge as the understanding of the research problem increases (Eisenhardt 1989b). To this end, case studies make use of a variety of data collection methods and sources (Yin 2009), such as interviews, observations, documents, physical artefacts and archival sources. The use of multiple sources facilitates the verification of evidence, which increases increasing the validity of resulting constructs and conceptualisation (Gibbert et al. 2008). Therefore, striving for the breadth and depth of evidence is necessary for research to be successful.

The method of corpus construction provides guidance on how to approach the construction of a research database. The process of construction is a cyclical and iterative method of data collection (Bauer & Gaskell 2000), and it resembles the open-ended data collection approaches advocated by Eisenhardt (1989b) and Yin (2009). The notion of corpus construction is borrowed into social science from linguistics, and the word corpus translates to “body” in English. While some linguistic corpora can be constructed to serve general purposes, the social

and organisation research typically rely on topic-based corpora which are constructed to serve specific purposes. This means that they are often thematically unified and narrow in scope while exposing a particular viewpoint or topic in a holistic manner. Bauer and Gaskell (2000) describe a simple stepwise procedure to characterise corpus construction. The iterative and cyclical approach includes phases such as preliminary selection of data, the examination of the variety in data and the subsequent extension of the database if and when this is needed. This process is repeated iteratively until the point of theoretical saturation is reached (Glaser & Strauss 1967).

In this research, data collection and analysis were carried out in parallel as cyclical, iterative and overlapping processes. Starting with data analysis before finishing with data collection allowed the researcher to respond to emerging themes and incorporate new sources of evidence into the analysis (Eisenhardt 1989b).

4.6.1 Documents as research data

The research database consists primarily of documentary evidence that is collected from public sources. The salient characteristics of documentary evidence and its use in qualitative and interpretative research are discussed below.

Qualitative research and theorising often builds upon data which is brought about by a researcher (Yin 2009). To elicit data, researchers rely on some of the various methods of fieldwork, such as interviews, focus groups or observation. In the field of information systems, these methods form an inseparable part of the research tradition. This is not surprising since the phenomena of interest (Avgerou 2000) are typically deeply embedded in organisational work settings and practices, which reside inside organisational boundaries. Data on such matters is rarely floating around freely in the public domain. Instead, it requires effort from researchers' part to collect and bring necessary facts to daylight.

Documentary evidence provides an alternative source of data. Silverman (2015) defines documentary evidence as "naturally" occurring information *"which have become recorded without the intervention of a researcher"* (p. 276). This

includes printed and electronic documents, blog posts, emails and other forms of digital traces that can serve as documentary evidence. These sources should not be overlooked as they may provide a researcher with rich and informative sets of data, which are readily available for analysis. In particular, harvesting data on digital ecosystems, platforms and open-source software projects appear to provide fruitful research avenues, and there are several examples of successful use of digital traces and documentary evidence. For example, Eaton et al. (2015) analyse a series of web blog entries to study the evolution of boundary resources on the iOS ecosystem. Similarly, digital traces and documentary evidence have been used to study Wikipedia editing patterns (Aaltonen & Lanzara 2015) and the coordination processes of the Linux kernel development (Shaikh & Henfridsson 2017). As these contexts of research transcend traditional organisational boundaries, much of related documents and data can be found form the sources that are publicly available.

While both elicitation and harvesting approaches to data collection can produce good results, there are some important differences. Starting from the origin of evidence, in the researcher led process of elicitation, a researcher typically defines the aims and scope of data collection based on their research interests (Yin 2009). Depending on the degree of open-endedness of data elicitation protocols, researchers are predisposed to impose a particular framing which serves as a filter during the process of data collection. On the other hand, with documentary evidence, a researcher has no control over the data generation process. Instead, the process through which evidence is generated, shared and put to use depends on the social and organisational arrangements against which the generation of evidence unfolds (Bowen 2009). This indicates that the filter which frames empirical evidence resides within the arrangements that give rise to data, meaning that documentary evidence cannot be considered as unfiltered and intrinsically true set of facts. Documents do not speak for themselves (Silverman 2015), and to mitigate against biases, a researcher must be cognisant of the social arrangements and processes through which the data that is used as evidence was created.

As digital environments record digital traces and abound with documents, it is often relatively easy to collect a large body of data. However, not all of them

carry equal importance in the light of research goals (Bauer & Gaskell 2000), and when a body of evidence increases in size and detail, it becomes increasingly challenging to find the proverbial needle in a haystack. To ensure that a body of documentary evidence is representative but manageable, a researcher must decide what to include and what to leave out of a research database. The next sections describe the data collection process and choices made during the data collection.

4.6.2 Construction of the research database

The research database consists primarily of the documents that have been collected from public sources. The database includes ROS related documents, blog entries, conference recordings, magazine and academic publications and emails from the ROS community mailing lists and discussion threads from online forums. These are complemented by the field notes from non-participatory observation and four semiformal interviews. The process and rationale of data collection are presented below, and the sources and categorisation of documentary evidence are listed in Appendices A to H.

The construction of the research database started during the pilot study phase before settling with ROS as a case study. Given the multidimensional and multidisciplinary character of the field of robotics, establishing clear boundaries between data collection and analysis proved challenging. Much of the pilot study phase focused on becoming familiar with the typical activities and language used in the field, as without an understanding of the language, concepts and research challenges, it was difficult to make sense of any empirical evidence. The pilot study phase involved attending robotics and artificial intelligence conferences, visiting research laboratories, interviews and a week-long summer school on field robotics with approximately 40 PhD students and academics. The events attended are documented in Appendix H. Subsequently, the resulting field notes and interviews were analysed, and the focus of research was placed on ROS as it was considered to offer a representative overview of the state of affairs as well as a rich body of empirical evidence.

A more detailed examination of ROS began by participating in a two-days ROS workshop and programming and running a virtual robot turtle in a simulated

environment (O'Kane 2014). In addition, attending the ROS conference in 2015 and robot software specific workshops in the European Robotics Conferences offered valuable insights into systems engineering and software development practices. These insights were documented in the field notes, and while this body of evidence was interesting and rich in detail, much of the evidence was too scattered to provide a coherent picture of the state of affairs.

To obtain a structure, representative and transparent body of evidence to increase reliability and generalisability, the focus of data collection and analysis was shifted on the publicly available documents. ROS is a vibrant open-source community, and there is a sizeable body of data publicly available. At the beginning, the collection and analysis of documentary evidence revolved around ROS as a technical artefact. However, the analysis soon revealed a rich historical and organisational dimension, which came to expand the scope of data collection. The early origins of ROS were traced back to the Stanford Artificial Intelligence Robot (STAIR) and Personal Robotics (PR) projects at Stanford University in 2005, and from there to Willow Garage where ROS was created along the PR2 hardware platform and made freely available as open-source software. Right from the beginning, ROS gained traction in the field. The stewardship of ROS was transferred to the Open Source Robotics Foundation (OSRF) in 2012 as Willow Garage ceased its active operations. Around that time, the ROS-Industrial (ROS-I) consortium was also founded to take ROS into industrial environments and the yearly ROSCon, a ROS developer conference, series was started. As an open-source framework and community, ROS is far from a monolith, sprawling developing branches to various directions, and various other events and trails of evidence could have been followed further.

Much of data was gathered from the communication channels and archives of the ROS community. The latest news, software releases, upcoming events and other matters of importance were communicated through the community channels and discussed in conferences and online forums. The documentary evidence stored in the research database came to include blog posts, conference recordings, emails and discussion threads as well as magazine and academic publications. These sources of evidence were considered as informative and well-consolidated sources, which could offer a representative view of the

organising logic of complex digital innovation. Table 2 summarises the contents of the research database in terms of their sources, types and description, and Appendix A provides a more detailed listing of data sources.

Source	Document types	Description
Stanford Artificial Intelligence Robot Project (STAIR) at Stanford University	conference papers (3), news articles (2), website (1), video recordings (3), grant application (1)	The STAIR project started at Stanford in 2005 and the origins of ROS can be found in the Switchyard software. The documentary evidence from the period 2005 to 2009 includes conference papers, web articles, the project website, a grant application and video recordings.
Personal Robotics Programme (PR) at Stanford University	conference papers (1), website (1), video recordings (4)	The PR project started at Stanford in 2005 to develop a hardware platform for mobile manipulation purposes. The documentary evidence contains a conference paper, website and video recordings.
Willow Garage	blog entries (407), news articles (8), website (1), video recordings (5)	Willow Garage's personal robotics programme continued the work started in the Switchyard and PR projects by developing hardware (PR2) and software platforms (ROS). The documentary evidence covers years 2007 to 2014 and includes blog entries by Willow Garage, news articles, website and video recordings.
Robot Operating System (ROS)	blog entries (1053), conferences papers (3), website, ROS wiki, news articles (18), email archives (2), ROS discussion forums (2)	ROS is one of the main outcomes of Willow Garage's personal robotics programme. It was moved to its own domain at ros.org in 2009. The documentary evidence covers years 2009 to 2017, including blog posts, conference papers, different ROS related websites, news articles, emails and messages from discussion forums.
Open Sources Robotics Foundation (OSRF)	blog entries (176)	As Willow Garage ceased its active operations, the stewardship of ROS was moved to OSRF. The documentary evidence includes blog entries and covers years 2012 to 2017.

Table 2: Summary of research database (continues next page)

Source	Document types	Description
ROS-Industrial (ROS-I)	blog entries (154)	ROS-Industrial consortia develop and promotes ROS for industrial purposes. The documentary evidence includes blog entries and covers years 2012 to 2017.
ROSCon developer conferences	recorded conference presentations (122, approximately 50 hours)	ROSCon is a yearly two-day conference for ROS developers. The documentary evidence in the form of conference presentations covers years 2012 to 2016, and includes a variety of ROS, OSRF, ROS-I and ROS2 related topics.
Robot Operating System – 2nd generation ROS (ROS2)	messages on design discussion (1155), ROS 2 design website (1)	The discussion concerning the second-generation ROS started in 2012. The messages regarding the future design requirements cover years 2012 to 2017 and the outcomes are document in the design website.
Observation and field notes	field notes and observation from ROS and robotics related events (12)	This section covers observations and discussions from different events that are documented in field notes. Includes ROS training, workshops and ROSCon participation among others. See Appendix H.
Interviews	semiformal and open-ended interviews (4)	Four semiformal interviews were carried out with robotics researchers on the general matters of robotics research and robot system development.

Table 2: Summary of research database

In the view of the saturation principle (Glaser & Strauss 1967), including additional sources of evidence to the research database would have produced only marginal benefits in the view of the research goals. Also, given the time and resource limitations, it would not have been feasible to include them into the analysis in any detailed manner. However, given the open-ended procedure of data collection, occasionally data from other sources was opportunistically incorporated into the research database when it was considered necessary and feasible, this being the case in particular with academic and magazine publications. This way, while the open-ended approach expanded the scope of data collection and analysis, it also resulted in a richer set of data that provided

important insights and enabled the triangulation of empirical findings. Tracing the contours of the history of ROS from 2005 onwards provided a comprehensive picture of the aims, developments and outcomes under different organisational auspices, contributing to the grounded understanding of the organisational and technological factors that underlie the organising logic of complex digital innovation.

As an administrative note, the research database was managed and analysed using Atlas.ti, which is a software package that is designed to support qualitative data analysis. It supports a variety of document formats, such as audio and video recordings and pdf documents. It also provides fucntionality to keep track of the progress and results of the analysis.

4.6.3 Evaluation of evidence

Data collection produced a sizeable amount of documentary evidence. Considering that the documentary evidence had been created without the researcher's intervention and for the purposes other than this research (Bryman 2015), its reliability and validity will be examined in the light of authenticity, credibility, representativeness and meaning (Scott 2014).

Authenticity refers to the authorship and provenance of documents to ensure they do not originate from dubious and misleading sources, whereas credibility refers to reliability and a degree to which documents are free from error and distortion. Representativeness, in turn, concerns to what extent any particular document is a typical representative of the phenomenon being investigated. Finally, the meaning of documents deals with whether the evidence contained in documents is clear and understandable to the researcher.

The documents collected into the research database are considered as authentic and credible. They originate from an open-source community and academic sources, and there is no apparent reason for the misattribution of the origin of a document. Similarly, there is no apparent reason to assume that the documents would not be credible. It is well possible that they may contain some errors but considering the constant scrutiny from the wider open-source community, reliability is expected to be relatively high. Concerning the representativeness of

the documents, things are more diffuse. While the body of documentary evidence can be viewed to represent well the case in general, individual documents on their own cannot be viewed as fully representative. The reason for this is that each document presents a particular piece of information providing a narrow and focused view onto some specific matter or event at a given point in time. As a final point, efforts were made to obtain a sufficient understanding of the state of affairs for ensuring the meaningfulness of documentary evidence.

The representativeness of documents can be reflected further in the view of their original purpose by asking questions such as why a document was created, whom it is targeted for or who authored it (Bowen 2009). Starting from the blog entries, they were collected from the Willow Garage, ROS, ROS-Industrial and OSRF websites. The entries were mostly created to communicate technical updates, latest news and organisational events to the members of the community, thereby providing a one-way communication channel. While each of the blog entries provides a piecemeal, selective and incomplete picture of the overall state of affairs, together they form a body of evidence based on which a representative picture of the overall course and timeline of events can be constructed.

In addition, the ROSCon conference presentations offer in-depth insights into topics that are relevant to the community. The presentations cover a variety of topics, such as the basics of ROS, key components and functionalities as well as their applications in different domains, contexts and task-specific use cases. In addition, the presentations are screened, peer-reviewed and selected by the organising committee and presented in front of a critical audience. The acceptance rate is around 30%. Therefore, the conference presentations are seen to offer a reliable and representative picture of the topics that are pertinent to the community. On the grounds of reliability and representativeness, they were given a prominent role in the data analysis and reporting of the results.

Given the amount of evidence and limited resources to analyse it, not all documents in the research database were given an equal priority. Instead, different types of documents were used to serve different research objectives.

The case description was constructed primarily from blog entries and journal and magazine publications, whereas the principal and the operative research questions were approached for the most part through the ROSCon presentations. Other material in the research database, the field notes, academic and magazine publications and other documentation, provided supporting evidence for triangulation and detailed queries throughout the research project. Detailed keyword-based queries were carried out when further clarifications were needed in some particular matters.

The simultaneous use of different types of documentary evidence from several sources established a well-rounded and triangulated view providing a solid and reliable foundation for the empirical and conceptual findings. The process and phases of data analysis are described next in more detail.

4.7 Data analysis

The process of thematic data analysis proceeded iteratively and in parallel with data collection to retain interpretative flexibility and ability to adjust to unforeseen avenues of research (Eisenhardt 1989b; Yin 2009; Walsham 1995). As per the process of thematic analysis outlined in Section 2 (4.2) (Bryman 2015), five different phases followed and overlapped each other as the project gradually unfolded from the familiarisation with topic and evidence towards empirical and conceptual findings. These phases are summarised in Table 3.

Phase	**Description**
Familiarisation	Learning the field and language.
Open coding	Label data into a list of embedded units of analysis as per the operative research questions.
Categorisation	Analyse embedded units of analysis and arrange them into categories as per recurring themes.
Thematisation	Analyse and abstract categories into themes.
Conceptualisation	Develop and conceptualise the links and connections between different themes.

Table 3: The five phases of thematic analysis after Bryman (2015)

The first phase focused on getting familiar with ROS and learning the language and central concepts used in the ROS community and in the field of robots and autonomous systems in general. This was followed by the second phase that entailed the first round of close reading and open coding of the documentary evidence. The first two phases provided an overall picture and understanding of ROS and related organisational arrangements and innovation dynamics. The main outcome of these two phases of analysis is the case description which is presented in Chapter 5.

Subsequently, the third phase rearranged and categorised the highly descriptive codes developed in the first round of open coding into more abstract higher-level categories in the view of subsystems, combinations and their characteristics. These categories were then further described and examined to elucidate their salient characteristics. In the fourth phase, these high-level categories were grouped into corresponding high-level themes. Finally, in the fifth phase, the relationships among different themes were further elaborated and incorporated into conceptual models that describe structural and functional characteristics of robot systems and shed light on the organising logic of complex digital innovation. The outcomes of these two phases are presented in Chapter 6.

The sections below describe the process of data analysis in more detail. The process and motivation to construct the case description are briefly described, before presenting the process of thematic analysis in more detail.

4.7.1 Case description

The first two phases of the analysis, the familiarisation with the topic and the first round of coding, produced the case description and provided the groundwork for further analysis. The case description provides an overview of ROS, its origins and evolution under different organisational settings. To date, while ROS is well-known among roboticists, it has escaped the attention of digital innovation, management and organisation researchers. Although there are papers and magazine articles written about ROS, they are often written from a technological point of view of software engineering (Quigley et al. 2009) or focus on the legacy of Willow Garage (e.g. Cousins 2014). Therefore, the

purpose of the case description presented here is to provide a broader view on the organisational and socio-technical aspects of ROS. To this end, ROS is presented as a communication system, open-source community and software development framework while also documenting its origins, evolution and expansion over time.

The case description proceeds in a chronological order and reflects the unfolding of the most central organisational events. The description was constructed by analysing and weaving together evidence from multiple sources, such as different websites, blog posts, conference presentations and journal and magazine articles, each of them providing a small piece of evidence that reflects some state of affairs or event at a given point in time. As some of the events are overlapping, the chronological order is occasionally sidestepped in order to keep the narrative coherent.

The starting point of the history of ROS is set here in 2005. There are two prominent options when deciding on what counts as an appropriate starting point. The options are the PR and STAIR projects at Stanford University in 2005 and the release of the first version of ROS by Willow Garage in January 2010. The starting point was set in 2005 because some of the foundational design decisions regarding the underlying ROS architecture originate from the research projects at Stanford. Since 2005, ROS as an open-source software development framework and community has undergone phases of development and expanded under the auspices of different organisations. Summarising the ten years of efforts from numerous organisations and thousands of contributors necessarily leaves out many rich details and phenomena that would warrant further research. However, an attempt has been made to capture and present the most salient and central organisational events and arrangements with a sufficient level of detail.

Therefore, the case description is expected to provide a reader with an overall view of the wider context, the case and the backdrop against which the thematic analysis can be reflected upon. The case description is presented in Chapter 5.

4.7.2 Thematic analysis

The phases three, four and five of thematic analysis, that is, categorisation, thematisation and elaboration respectively, produced the findings that are reported as main outcomes of this work in Chapter 6.

The first two phases, familiarisation and first round of coding, produced the ground-work for the subsequent phases of analysis. The close reading and open coding were carried out using the operative research questions as sensitising devices in order to identify the instances of subsystems and combinations that would serve as embedded units of analysis. This resulted as hundreds of individual units of analysis, each of them representing some particular subsystem or combination. As the process of open coding was not very selective, the codes labelled a variety of subsystems, components, parts, combinations and related characteristics and other phenomena. They were close to data, highly descriptive and occasionally, although very interesting, not very relevant in the view of the overall objectives of this research. However, the open coding provided the necessary raw material and insights for further analysis.

Subsequently, as per the process of thematic analysis, categorisation began. To rearrange the initial codes into appropriate categories, the focus was placed on the salient characteristics of subsystems and combinations with an aim to identify recurrent patterns, similarities and differences among the embedded units of analysis. While the initial listing was comprehensive, it was not very tractable. The list was long and relationships among embedded different units of analysis unclear; it seemed that each of the units rendered a set of patterns that would have warranted various alternative categorisations. This was made more difficult by the fact it was sometimes challenging to separate an instance of a subsystem from an instance of a combination as they often represent the two sides of the same coin, especially when the purpose of a subsystem is to facilitate combination.

Initially, the boundary objects that are usually discussed in the context of digital innovation, such as standardised application programming interfaces, systems development toolkits, boundary resources (Eaton et al. 2015) or platforms in their different guises (de Reuver et al. 2017), were searched for. However, the

ROS framework proved highly heterogeneous, defying straightforward categorisation along the lines of usual boundary objects. Several alternative conceptual lenses were tried in the spirit of interrogative inquiry, yet none of them seemed to describe what was being observed.

Subsequently, a more grounded and data-driven approach was adopted. Instead of looking for specific organisational characteristics and combinatorial relationships in and among the listed objects and artefacts, the focus was placed on examining the purposes they serve and functionalities they produce. This was done by keeping an eye on Arthur's (2009) general conceptualisation of technologies as purposed systems that harness some effect or phenomenon, a conceptual basic principle. This provided a more fruitful avenue for categorising efforts, although it required the researcher to get familiar with a broad range of technologies with sufficient level of scientific, logical and engineering details. This was a convoluted and iterative process that overlapped with data collection, involving a non-trivial amount of exploration, additional clarifications and boundary-making to ensure that findings were reliable and relevant in the view of research objectives and current literature on digital innovation. In the end, the resulting functionally-oriented categorisation scheme reduced the number of categories to 15 and provided an adequate foundation for thematisation.

Even if the instances of subsystems and combinations served well as initial sensitising devices, their role changed over the course of research. As the iterative process gave rise to new conceptual categories, those new categories came to assume the role of the unit of analysis. At the same time, the initially listed subsystems, objects and artefacts of different sorts turned into representative instances of those categories. In a way, the initial embedded units of analysis were turned into units of explanation as they came to provide the facts and evidence to justify the proposed categorisation.

The fourth phase, thematisation, proved more straightforward. The 15 categories were abstracted further and recast as six abstract themes that represent domains of technologies that carry out different functionalities, have particular characteristics of combination and to an extent reside at different technological and architectural levels.

Then, the fifth phase, conceptualisation, examined relationships among different themes and categories and conceptualised them into models that characterise the structuring of complex digital innovation. Conceptualisation produced two outcomes. First, it produced a model that provides a structural-functional conceptualisation of complex digital innovation, which, in turn, was used as a lens to analyse data so that the primary research question could be approached and answered. This process produced a model that conceptualises a mode of systems development in the view of complex digital innovation.

The process of thematic analysis unfolded as an iterative and interrogative process (Hintikka 1999); it gradually proceeded towards the answering of the principal research question, relying on the initial operative research questions while continuously evolving working hypotheses until a satisfactory answer had been reached. This way, codes, categories, themes and conceptualisations emerged from data, albeit in a partially theory-driven way that constantly deliberated tentative findings in the light of current literature and theoretical framing presented in Chapters 2 and 3. The reporting of the findings is described in the next section.

4.7.3 Reporting of themes and categories

The purpose of describing the research design and methods is typically to provide a reader with a detailed exposition on how research findings were arrived at. However, it is not always feasible or practical to expose a full set of details, paths tried and discarded during the analytical process, especially when the process has been highly iterative and cyclical. Yet, even if the process of analysis could not be exposed in a detailed manner, a comprehensive presentation of research outcomes and supporting evidence can be provided to demonstrate and justify the resulting empirical and conceptual findings. With reference to Simon (1996, p.132), *"[s]olving a problem simply means representing it so as to make the solution transparent."*

Considering the iterative and exploratory character of this research, it is not feasible to expose the full details on how the categories and themes were arrived at beyond the general description of the main phases of the process. Therefore, efforts have been made to present the research outcomes along with the

supporting evidence so that the proposed answers to the research questions are transparent and empirical and conceptual findings are replicable and refutable.

The process of thematic analysis produced six themes and 15 categories. These themes were labelled as (1) robot systems, (2) physical embodiments, (3) communication systems, (4) transformation systems and (5) visualisation and testing systems, and (6) the ROS community and software development, and they are presented later in this order so as to construct a coherent narrative that gradually unfolds and explains a variety of subsystems and their relationships. These themes and findings provide the foundation for the conceptual development.

To link categories and themes explicitly back to evidence (Yin 2009), the ROSCon presentations were revisited in the light of the established themes and categories. Each presentation was re-examined and the subsystem that formed a focal point of the presentation was identified, according to which the presentation was then assigned to the theme it was considered to be a representative of. The results of this process are documented in Appendices B to G. Each of the appendices lists the presentations that belong to that particular theme. The listing includes an identification code, year, presentation title, duration and a subsystem that forms the focal point of a presentation and respective categorisation. When the themes and categories are presented and discussed in Chapter 6, the ROSCon presentations are cited as sources of evidence in the format (Appendix: Code).

Moreover, efforts have been made to expose data so as to provide a reader with clear illustration of the themes and categories discussed in particular sections. Links to additional sources of evidence are also provided in footnotes when necessary. The purpose of this is to make evidence and reasoning as accessible and transparent as possible to increase the validity of findings.

Table 4 summarises the distribution of themes over the ROSCon presentations. For each cell, the first number indicates the number of presentations by the year and theme. The second number inside the brackets sums duration in minutes. The rows are ordered by the total length in minutes devoted to each theme.

Measured this way, of the six themes, the theme of communication systems has gathered the largest share of attention. This is not surprising since the communication system forms a central part of the functionality of ROS. The theme of the ROS community and software development came second, whereas the theme of robot systems holds the third place. The fourth place goes to the theme of transformation systems, and the theme of visualisation and testing systems are on the fifth place.

Theme/Year	**2012**	**2013**	**2014**	**2015**	**2016**	**Total**
Communication system (B)	7 (231 mins.)	9 (142)	4 (109)	4 (123)	6 (142)	30 (747)
ROS community and software development (C)	5 (74)	8 (164)	4 (64)	7 (153)	6 (131)	30 (586)
Robot systems (D)	4 (118)	4 (76)	6 (165)	4 (115)	3 (79)	21 (553)
Transformation systems (E)	4 (170)	5 (129)	3 (117)	4 (101)	1 (19)	17 (536)
Visualisation and testing systems (F)	1 (46)	6 (120)	2 (51)	2 (32)	5 (123)	16 (372)
Physical embodiments (G)	- (-)	2 (24)	- (-)	2 (30)	4 (86)	8 (140)
Total	21 (639)	34 (655)	19 (506)	23 (553)	25 (580)	122 (2934)

Table 4: The distribution of themes across conference presentations

The last but not least is the theme of physical embodiments. In the end, while very few presentations focus primarily on the embodiments, they are implicitly present in nearly all presentations. Also, while the presentations were assigned to different themes according to their primary focus, they frequently include elements from different thematic categories as different themes are highly interrelated. Occasionally an argument for an alternative assignment could have been made.

These themes, categories and their relationships are presented in Chapter 6. They are first summarised in Table 5 at the beginning of the chapter and then described and discussed in more detail throughout the rest of chapter

4.8 Summary

This chapter described the design and methodology of this research to identify and locate prominent subsystems and their characteristics of combination in order to examine how the tensions between the specificity of designs and distributedness of knowledge and control unfold in the ROS ecosystem. To that end, this research was designed as an embedded case study that follows the process of thematic analysis that is guided by theoretically-driven and tentative a priori concepts.

At the beginning of the chapter, case study research was described and discussed as an evolving inquiry, after which the method and process of thematic analysis and the role and function of tentative a priori conceptualisations in this research were described. Subsequently, the design of this research was presented to establish the overall structure of the work. Then, the process and rationale of case selection were described to draw boundaries around the scope of research and potential knowledge claims. After that, the construction of the research database along with the characteristics of documentary evidence were described. Finally, the chapter closed by describing the process of thematic analysis which produced the case description and the themes and categories for conceptualising the unfolding of the tensions between the integrality of designs and distributedness of knowledge and control.

5 Case description

This chapter introduces ROS and describes its history and salient characteristics in order to provide a general overview of the case and the wider context it is embedded in. As mentioned earlier, ROS stands for the Robot Operating System (Quigley et al. 2009). ROS is a widely used software development framework in the field of robotics and, as an open-source community, it brings together a variety of robot software developers, users and contributors, from academia and industry.

Over the last ten years, ROS has undergone a significant expansion and transformed from a simple communication library to a widely adopted software development framework and open-source ecosystem. To present ROS and its development paths to date, this chapter, as described in the previous chapter, arranges the collected documentary evidence into a narrative which outlines the central events from organisational and technical viewpoints to illustrate the stated objectives and (un)planned outcomes of ROS development. In other words, it is a narrative that outlines how two university-based research projects combined with an ambitious and well-financed vision initiated a chain of events that turned out as a global open-source community.

This chapter unfolds as follows. The prominent characteristics of ROS are presented first to provide an overview of ROS as a software development framework that was created to support collaborative development of software for robots and autonomous systems. After that, central organisational events are presented. The ten years history of ROS is divided into three phases and presented over three sections. The first phase centres on the Stanford Artificial Intelligence Robot (STAIR) and Personal Robotics (PR) projects at Stanford University between 2005 and 2009. The second phase covers years from 2007 to 2014 under the auspices of Willow Garage, a well-funded research-oriented start-up, and how the technologies conceived in the STAIR and PR projects were developed further and made publicly available as ROS and PR2. The third phase begins in 2012 when the Open Source Robotics Foundation (OSRF) was founded, ROS-Industrial consortia set up and the yearly ROS software developer conference (ROSCon) launched, marking the wider uptake and

institutional support for ROS. After the presentation and discussion of the central organisational events, the penultimate section focuses more on the technical side of ROS and discusses the ongoing development of ROS2. After that, the concluding section summarises the main points.

5.1 Three viewpoints on ROS

This section presents ROS from three different points of view to outline its basic characteristics. Regardless of the name, ROS is not an operating system in the traditional sense of the word. Whereas Linux, Windows and macOS which serve as a centralised layer of abstraction, control and scheduling between the computer hardware and applications that run on it (Tanenbaum & Bos 2014), ROS is better understood as a communication system, open-source community and software development framework.

To begin with the communication aspect, perhaps the best place to start unpacking ROS is to appreciate that software architectures that control robot systems are typically highly-distributed. To exemplify, Figure 6 illustrates the distributed computational arrangement that was used in the "fetch-a-stapler" demonstration created by the STAIR project (Quigley et al. 2007). Much of what follows is related to this arrangement in a way or another. In the language of ROS, this network of computations is known as the ROS graph, and it consists of two kinds of elements, computational processes and their interconnections. The graph conceptualises the software of a robot system as a set of distributed computational processes, and the core functionality of ROS as a technological artefact is to establish and manage the interconnections among computational processes. In addition, it is worth to the note that the overall architecture of other robot software development frameworks, such as Orocos (Bruyninckx 2001) and YARP (Metta et al. 2006), also similarly adhere to this type of distributed scheme of computation.

As an open-source community, ROS brings together a broad range of roboticists from academia and industry. In this community, users and contributors share robotics-related knowledge and software with each other, and while ROS is to a large extent a community effort, OSRF has a central role in the coordination and facilitation of ROS development.

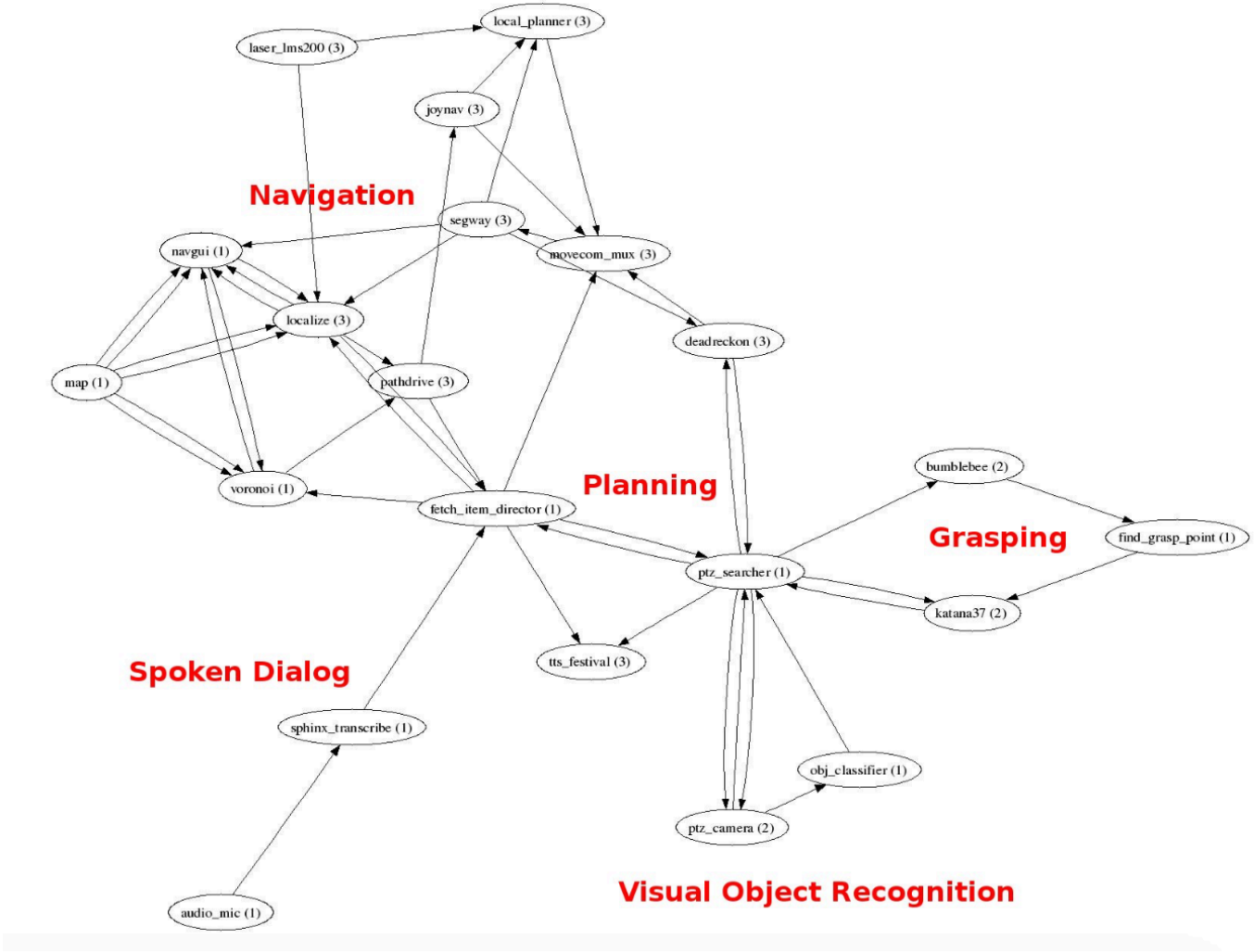

Figure 6: The computational graph used in the "fetch a stapler" demonstration (Quigley et al. 2007)[7]

[7] Reprinted from STAIR: Hardware and Software Architecture, by Morgan Quigley, Eric Berger and Andrew Y. Ng, AAAI 2007 Robotics Workshop, with permission.

OSRF develops and maintains the core functionality, central components and development tools of ROS and coordinates and manages the yearly releases, which distribute a stable set of the core functionality and central components. In addition, OSRF coordinates and supports collaboration within the ROS community and develops links to wider industrial ecosystems. To make all this possible, OSRF employs a core development team, maintains a build system and communication channels to distribute software and related knowledge within the community. The centralised build system provides the ROS community with infrastructure and methods for centralised software distribution, whereas different communication channels, such as the websites ros.org, wiki.ros.org and answers.ros.org, discussion forums and mailing lists, facilitate knowledge sharing and discussion on various ROS related matters. The combination of the ROS communication system, software components, libraries and development tools that are shared and created by the community renders ROS as a framework that supports collaborative software development.

While OSRF develops the ROS communication system, selected core functionalities and tools and methods for software developers to establish interconnections between distributed computational processes, a variety of software packages that produce different computational processes originate from the members of community, such as universities, research institutions and hardware vendors and other organisations. In the context of ROS, the concept of a software package refers to a unit of code sharing, and it can contain either a small and simple piece of software or large and complex functional module. Although OSRF maintains the centralised software build infrastructure that facilitates the distribution and reuse of software packages, this infrastructure, however, is not used to host the package-specific source code. Instead, it retrieves source code from different organisational code repositories if organisations package their software in a ROS compatible manner and wish to have it shared through the ROS infrastructure. This way, the organisations that originate code can retain the control over it even if they are willing to share as open-source. As a software development framework, ROS provides developers with a collection of open-source software libraries, tools, infrastructure and shared practices which are designed to support collaborative development of organisationally and computationally distributed software development.

Therefore, ROS can be approached and analysed from different viewpoints. First, it can be seen as a digital communications system which is used to develop robot software as sets of interconnected computational processes. Second, it can be seen as a community which brings together a variety of user and contributors from academia and industry that share the interest in the development of robot systems and related functionality. Third, it can be seen as a framework, as a collection of software that facilitates the construction of systems which are composed of a wide array of advanced computing technologies.

While each of these viewpoints offers a different picture, they point towards the same object. They form an intricate web of technological and organisational relationships, which intertwine at different technological and organisational levels. In this light, ROS could be seen as a clan innovation network (Lyytinen et al. 2015). In clan innovation networks, actors share an interest in a specific product type and concept, have access to a common set of tools and have complementary and overlapping knowledge while also sharing a common vocabulary, yet they are not bound by any centralised and hierarchical control. The next section presents the early origins of ROS and how it emerged from the two research projects at Stanford University.

5.2 PR and Switchyard at Stanford

The early origins of ROS can be traced back to the Personal Robotics (PR) and Stanford Artificial Intelligence Robot (STAIR) research projects at Stanford University around 2005. The two projects focused on different areas of robotics technologies but shared an overall goal of advancing mobile manipulation in human environments, such as homes and offices. Whereas the PR project sought to develop a safe and robust physical robot platform that could move around and manipulate objects, the STAIR project aimed at bringing different artificial intelligence technologies together in order to develop software that is capable of carrying out tasks autonomously in human environments. Therefore, the PR project focused primarily on physical embodiments and robustness while the main interest of the STAIR project was in computable matters. Together they represented the physical and digital aspects of the development of robots and autonomous systems.

The PR project developed the PR1 prototype hardware platform for mobile manipulation tasks (see Figure 7), and it is a predecessor of the PR2 platform, which came to provide the first reference hardware platform for ROS. In 2005, the hardware platforms capable of carrying out mobile manipulation tasks were in short supply. As a response, the PR project set out to develop a platform that would be capable of moving around and manipulating objects in human environments in a safe and reliable manner (Wyrobek et al. 2008). The objective was to provide software developers with a robust robot platform upon which more sophisticated functionalities could be built. The development of a robust hardware platform was seen as analogous to the development of computer hardware. It would provide software developers with a stable foundation to build on; software developers could leverage underlying hardware capabilities and proceed with application development without needing to worry about the underlying hardware or the associated low-level software that controls and protects the hardware.

The PR1 hardware was designed together with the low-level control software. The purpose of this was to create a layer of abstraction for separating the hardware from higher-level operational commands. This abstraction would make software development easier as developers could control the hardware through the higher-level operational commands that would be converted to specific actuations and movements by the lower-level control software. A developer could simply command a mobile base to drive to a certain direction at a certain speed without needing to know about the implementation details of the wheels, such as their locations, geometry, steerability or drivability. In addition to providing a layer of abstraction, the low-level control software would also protect the robot hardware by preventing the execution of harmful commands, the commands that would not conform to the physical properties of the robot or were potentially caused by bugs or faulty computational models.

Overall, the introduction of a hardware platform with the associated low-level control software was seen as a way to shift the focus of development from the mechanical and electrical engineering to software engineering and application development. Leveraging common hardware, useful applications could be developed faster to serve a variety of use cases and market needs.

Figure 7: Personal robot PR1 prototype
(Wyrobek et al. 2008)[8]

Whereas the PR project focused on hardware and low-level control software, the STAIR project focused on higher-level control software. The STAIR project set out to develop computational models, to bring together and developed further knowledge and technologies of different domains of artificial intelligence (Ng et al. 2008; Ng & Khatib 2006). The plan was to integrate and implement different artificial intelligence technologies into physical robot platforms for research and teaching purposes, and thereby pave the way towards practical applications and advancement of personal robotics. The project would provide a framework within which different artificial intelligence technologies could be improved

[8] Reprinted from 2008 IEEE International Conference on Robotics and Automation, Towards a Personal Robotics Development Platform: Rationale and Design of an Intrinsically Safe Personal Robot, by Keenan A. Wyrobek, Eric H. Berger, H.F. Machiel Van der Loos and J. Kenneth Salisbury, copyright (2008) IEEE.

while also simultaneously bringing them together to compose more sophisticated and capable models of autonomous behaviour.

The request “fetch a stapler” was used to demonstrate the need to combine computational processes from different domains of artificial intelligence (Quigley et al. 2007). To fulfil the request, a robot needs to be able to record a spoken command, recognise the meaning of the command, navigate through office rooms, corridors and doors in the search of a set of features that matches the representation of a stapler, grasp it, lift it up and bring it back to the location where the command was first voiced. This seemingly simple task demonstrates the spectrum of technologies and capabilities that need to be mastered and integrated in order to produce a complete behaviour; fetching a stapler requires technologies and skills on areas such as voice recognition, message parsing, image and object recognition, mapping, localisation, navigation, path, motion and grasp planning in three-dimensional spaces over time. Given the associated intricacies, each of these areas require deep and specialised knowledge on what is being computed for and how to compute it, yet the production of behavioural models requires an orderly integration of a wide array of advanced computational technologies and specialist knowledge, something which is scattered across a variety of research domains and organisations. Therefore, from the technological and organisational point of view, fetching a stapler is a formidable integration challenge.

The STAIR project used two different mobile manipulation platforms as research hardware. They differed in terms of their frame, sensors, mobile base, arms, grippers, size and geometry, and they underwent constant change throughout the project as teams of students and researchers made changes to the robot and computing hardware and software applications. The hardware platforms are pictured in Figures 8 and 9.

This technological and organisational complexity posed several integration challenges. What was needed was a flexible software development framework that could handle both parallel and distributed computing and software development as well as heterogeneous hardware, computing and software environments (Quigley et al. 2007). The framework was expected to facilitate

distributed and parallel computing since for example motor commands and longer-term planning functions tend to unfold in different time scales and may run on different computers to ensure the instant availability of computing capability. Also, the framework could not be tied to any particular computer operating system as computing environments were heterogeneous. Similarly, it could not be tied to any specific robot hardware or components as the project had two different and constantly changing robot hardware platforms in use. Finally, it was expected that that framework supports the separation and combination of functional software components in order to allow multiple teams to develop and test their code in parallel.

Figure 8: The STAIR 1 robot (Quigley et al. 2007)[9]

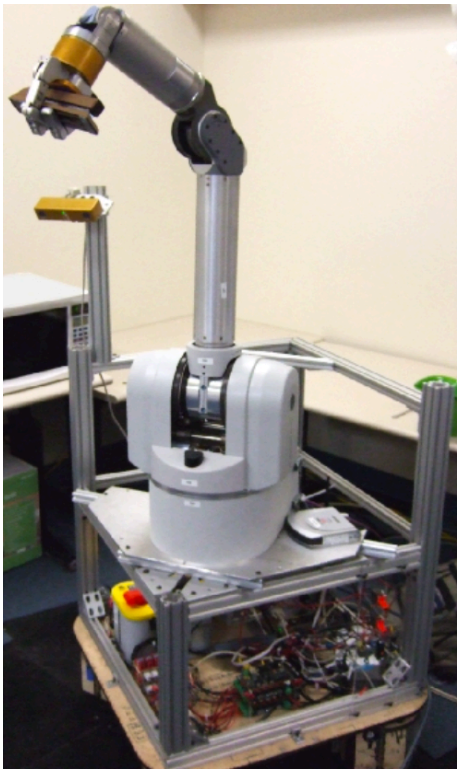

Figure 9: The STAIR 2 robot (Quigley et al. 2007)[9]

After evaluating different design approaches (Quigley et al. 2007), the communications framework and library called Switchyard was developed to fulfil these requirements. Switchyard offered a method to connect various software components so that a robot software could be run as a group of interconnected computational processes, that is, as a virtual and distributed cluster of processes that operates on top of some underlying group of networked

[9] Reprinted from STAIR: Hardware and Software Architecture, by Morgan Quigley, Eric Berger and Andrew Y. Ng, AAAI 2007 Robotics Workshop, with permission.

computers. Figure 6 illustrates this concept by showing the computational graph that was used in the “fetch a stapler“ demonstration. The nodes in the graph represent individual computational processes whereas the arrows represent the interconnections between the processes. The large texts show the functional tasks, such as spoken dialogue, navigation, planning, visual object recognition and grasping, performed by different regions (Quigley et al. 2007).

This distributed, loosely-coupled and flexible software architecture appeared to provide a solution to the technical and organisational challenges faced in the distributed development of distributed systems. The lessons learned from Switchyard were taken on board at Willow Garage as the company embarked on its personal robotics programme.

Willow Garage’s personal robotics programme picked up the work started at Stanford in the PR and STAIR projects and set out to develop PR2, a robot hardware platform for research purposes, and ROS, the accompanying software development framework. This endeavour is discussed in the next section.

5.3 PR2 and ROS at Willow Garage

Willow Garage was founded in autumn 2006 in Menlo Park, California. It started out as a well-funded private research laboratory that focused on creating the next generation of robotic devices. The company stated it had resources to maintain a research lab of 60 people indefinitely (Cousins 2014), emphasising its willingness to risk-taking and experimentation in order to pursue cross-disciplinary innovation without committing to any specific timeline. At the beginning, the company pursued three different domains of robotics, namely, autonomous cars, boats and personal robotics, before directing its efforts to personal robotics.

In October 2007, the company announced its plans on personal robotics. The mission was to develop a new personal robotics platform that could serve people in human environments without a risk of causing serious injury. The endeavour was a continuation to the work initiated at Stanford, and the initial target was set to build ten robots for research purposes in a year’s time. To this end, the

company hired the graduate students from the PR project to lead the effort and started the development of PR2 (see Figure 10), an advanced version of PR1.

The development of ROS began at the same time in order to provide a software framework upon which advanced higher-level application software could be developed. In order to support distributed software development and computation, the design principles of ROS adhered to those of Switchyard. The development of ROS progressed in close collaboration with the STAIR project, and the company supported the STAIR project financially as well.

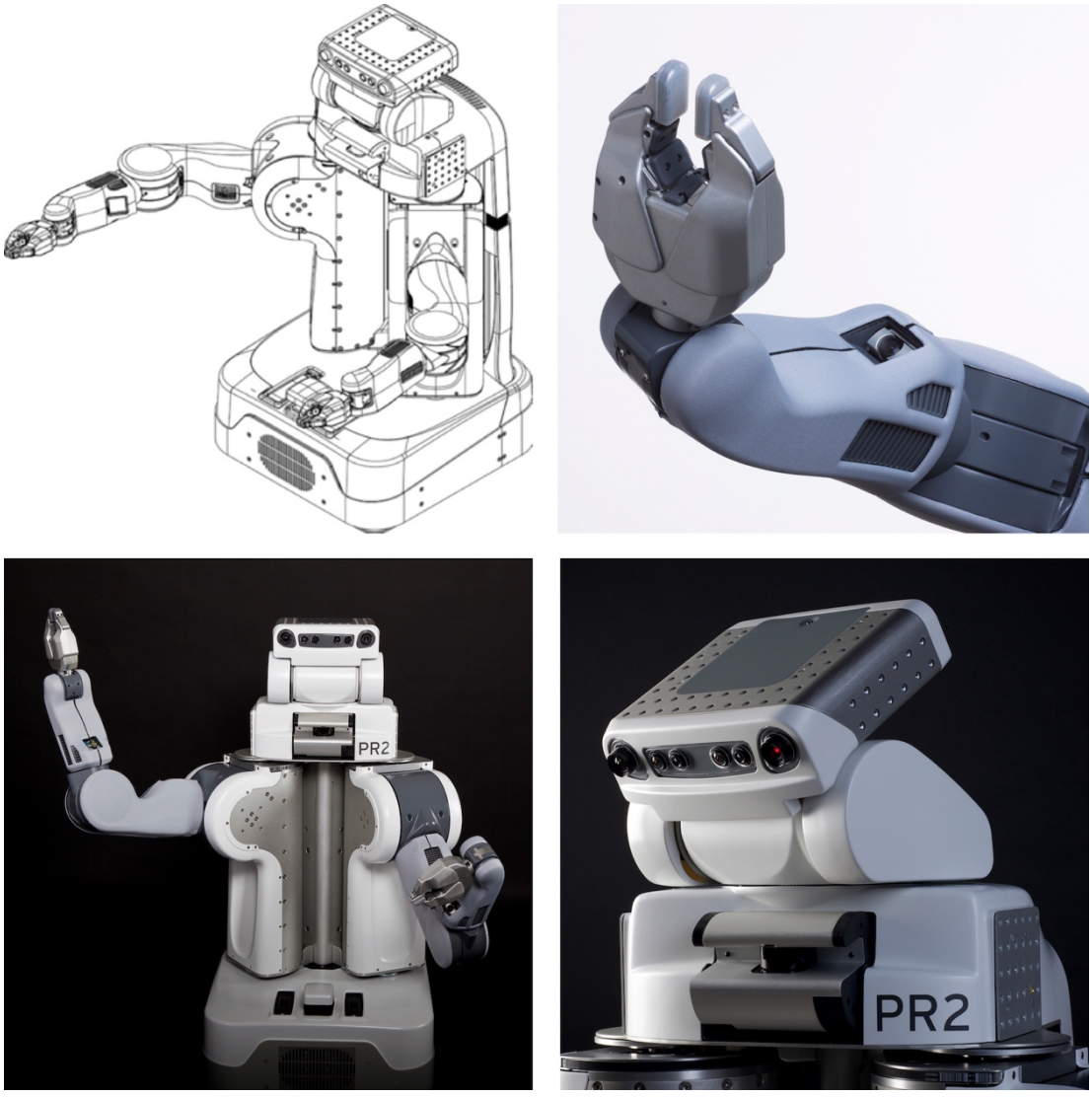

Figure 10: The PR2 robot[10]

[10] Republished from the press room of Willow Garage at willowgarage.com. Copyright (2008-2015) Willow Garage.

The company presented its vision of robotics platforms in various conferences and venues. The vision was demonstrated with a reference to the computer and smartphone industries where systems architectures are often characterised as layered stacks, hardware sitting on the bottom, an operating system on the top of the hardware and applications then on the top of the operating system, and these layers interact with each other through well-specified interfaces. Against this backdrop, PR2 was presented as a foundational hardware layer with open interfaces and standards, whereas ROS was pictured as a layer sitting on the top of the robot hardware, as presented in Figure 11 from the Willow Garage's presentation in the International Robot Exhibition (IREX) in 2009.

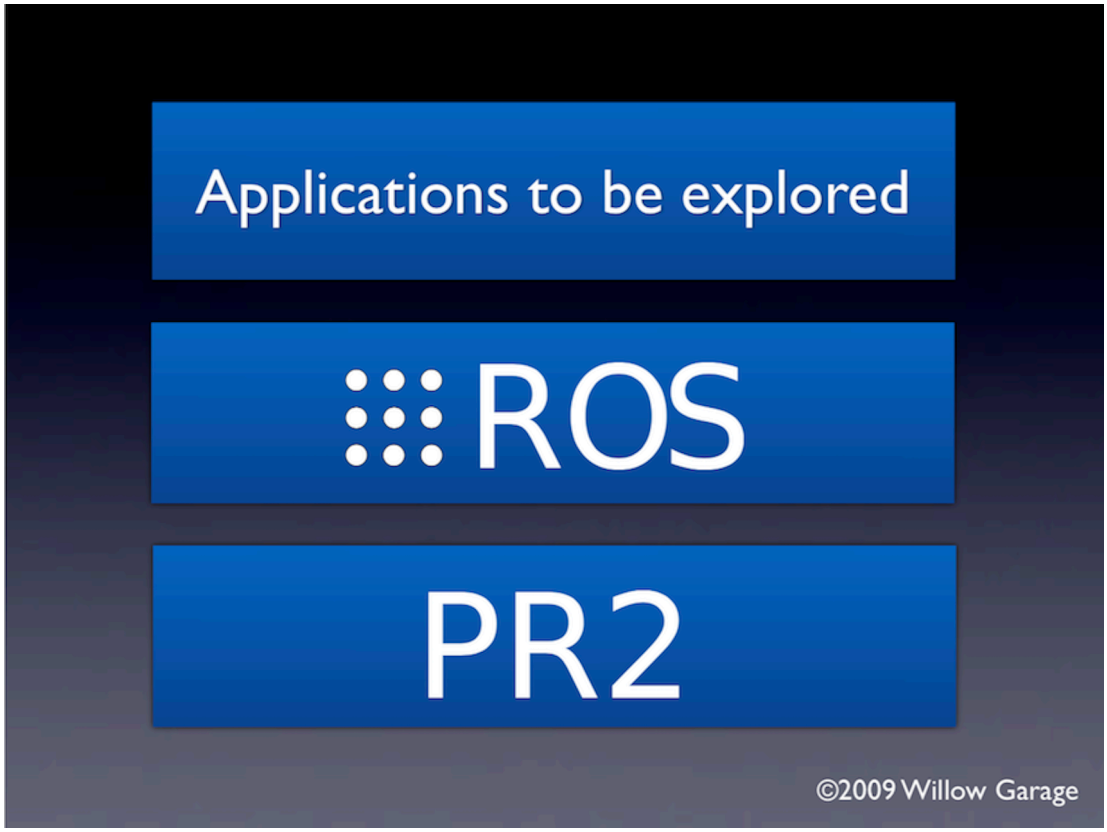

Figure 11: PR2, ROS and applications as a layered stack[11]

ROS was seen as a Linux-like environment for open-source robot software development. The company also encouraged other companies to build hardware platforms and join the ROS community. The project was occasionally envisaged

[11] Republished from Willow Garage's presentation in the International Robot Exhibition (IREX) in 2009. Copyright (2009) Willow Garage.

as a path towards robot-specific application stores where new functionalities for robots could be sold and bought.

To track the progress, the company divided the personal robotics programme into four milestones, the final one of them marking the customer delivery of ten operational PR2 robots with documented software. The first milestone of 3.14 km of uninterrupted indoor navigation in two days was achieved in December 2008. This tested hardware, electronics and software and demonstrated the level of integration and robustness across different layers, ranging from the low-level hardware control upwards to the higher-level software that performs mapping and navigation in two-dimensional spaces. The second milestone was reached in June 2009. This time the criteria included 42 km of indoor navigation and demonstrated PR2's capabilities in mobility and three-dimensional manipulation. The robot was required to open doors and go through the doorways and plug itself into electric sockets. The third milestone was achieved in January 2010. It focused on the stability and documentation of software to make the robot and its software usable and accessible to the researchers and developers outside Willow Garage. This way, PR2 came to provide a well-document reference implementation of ROS and its various components and their interactions.

During the period leading up to the third milestone, the ROS documentation was moved from the Willow Garage website into its own domain at ros.org in August 2009. The site consolidated knowledge and community efforts. Technical details and instructions were documented in the wiki, and a web blog was set up to communicate ongoing affairs along the mailing lists. This move was seen as strategically significant. It established ROS as a community effort that was not limited to the purposes and boundaries of Willow Garage regardless of the company's influential role in the introduction and development of ROS. While ROS was developed with reference to the PR2 hardware and to support distributed computation for mobile manipulation, it started attracting attention from the wider robotics community right from the beginning. The flexible and distributed architecture made it applicable to a variety of use cases. Also, as the source code of ROS was made publicly available and distributed under a permissive open-source license (BSD), it remained

freely modifiable and applicable for research and commercial purposes. At the time of the introduction of ros.org, eight open repositories containing ROS compatible code had already been created by entities external to Willow Garage. ROS was being extended and experimented on a variety of hardware platforms. The conference paper *ROS: an open-source Robot Operating System* (Quigley et al. 2009) was presented at the International Conference on Robotics and Automation (ICRA) in May 2009, and it is the seminal academic paper that presents the design principles of ROS.

With the initial one-year schedule stretched to nearly three years, the company announced the completion of the third milestone and the release of ROS 1.0, the first official version of ROS, in January 2010. At the same time, the company also announced a call for proposals to participate in the Beta Program. The proposals were expected to demonstrate ambitious and innovative research using PR2 as a research platform. Ten proposals would be awarded the PR2 platform worth of $400 000 for a two-year period free of charge including maintenance, technical support and travel support. Out of the 78 proposals received from universities and research laboratories, eleven were successful (Cousins 2010). In return, the Beta Program participants were required to release the software developed on PR2s to the wider community as open-source.

The eleven PR2s were ceremonially awarded to their recipients in the Graduation Party in front of the press and 300 guests in May 2010. The Willow Garage founder Scott Hassan presented his vision of service robotics as a new industrial revolution and emphasised the company's commitment to the next generation of robotics platforms and open source software. With platforms, robotics was envisaged to turn from a hardware problem to a software problem by facilitating modular, gradual and cumulative innovation and collaboration among researchers, developers and industry. Although the PR2 hardware platform was the most tangible and visible result of the development and engineering efforts, the representatives of Willow Garage were also very excited about the rapid uptake of ROS and the growth of the ROS community.

The final project milestone was achieved in June 2010 after all eleven PR2s had been shipped out to their recipients. Over time, PR2 proved to be a powerful

research platform and, in total, approximately 50 PR2s were delivered for research purposes. However, given the relatively high price, there was not much demand outside well-funded research laboratories and organisations.

In organisational terms, the personal robotics programme was a highly collaborative endeavour. Along the multidisciplinary group of Willow Garage researchers and engineers (60 at peak, 100 over time), the company hosted a number of visiting scholars and run an extensive internship programme. More than 100 interns and visiting scholars from high-profile robotics research laboratories worked from two to nine months at Willow Garage (Cousins 2014). They brought in the knowledge and technologies from their home institutions and contributed to the development of PR2 and ROS. This way, while the company contributed much to the open-source software development and community building, it also gained much by making use of various software components, development tools and algorithms that were first developed and made available elsewhere. In turn, when the visiting scholars and interns returned to their home institutions, they spread their knowledge on ROS expanding the community around ROS. The characteristics of ROS that facilitated distributed development were put to use on a global scale and across different domains of robotics.

Throughout the personal robotics programme, the company frequently released new and updated ROS distributions as it developed the core functionality, development tools and infrastructural capabilities. These were documented in the wiki.ros.org site and communicated to the wider community through the ros.org website, blog posts and video updates.

In 2011, after succeeding in the initial goal of developing a research platform capable of mobile manipulation, the company entered a new phase and turned its attention to the commercial application of its technologies and experiences. Although PR2 was successful as a research platform, its capabilities and robustness had not reached the level required to carry out a variety of tasks in open-ended and unstructured home and office environments. Consequently, different options were evaluated as business opportunities were searched for, and efforts were directed into more constrained and repetitive application and

task domains which could be fulfilled with special-purpose robots and applications. This search for opportunities did not result in a single preferred outcome. Instead, it produced a stream of spinouts as the founder and employees embarked on a variety of ventures. Redwood Robotics started developing cheaper robotic arms and was acquired by Google. Industrial Perception set to work on perception pipelines for three-dimensional vision and was acquired by Google as well. HiDoF embarked on ROS consulting and was also acquired by Google. Suitable Technologies proceeded to cater telepresence markets with telepresence robots. Unbounded Robotics (relaunched later as Fetch robotics) focused on warehouse logistics developing mobile manipulators and freight-carrying robots, whereas Savioke set out to cater to hoteliers by focusing on hotel room deliveries. The visions of general-purpose robotics platforms had been scaled down to purpose-specific applications.

The spin-offs were not limited to hardware and applications. A series of open-source software projects were also spun off. OpenCV, an open-source computer vision library, initially developed at Intel and later modernised and extended under the auspices of Willow Garage, was moved to the OpenCV Foundation. The Open Perception Foundation was founded to provide a home for Point Cloud Library (PCL), an open-source perception library for depth images, whereas the software for controlling robot bases and arms were consolidated into the MoveIt! motion planning platform. ROS itself found a new home at the Open Source Robotics Foundation (OSRF). The legacy of Willow Garage also lives in the TurtleBot hardware (Gerkey & Conley 2011), which is a cheap mobile base for educational purposes. It is now in its third edition and also serves as a reference implementation of ROS.

After a series of spin-offs, Willow Garage ceased its active operations in January 2014. In retrospect, Willow Garage's ambitions to speed up the development of the robotics industry can be seen either as a failure or a success story. Willow Garage did not succeed in introducing a general PC like hardware platform. That remained out of reach. However, the company further developed and established a software development framework and open-source community that appears to address some of the central technical and organisational integration challenges in a manner that the wider community sees beneficial.

ROS can be viewed as one of the most important and impactful legacies of Willow Garage in the field of robotics. The wider uptake of ROS and the changes in ROS stewardship are discussed next.

5.4 The wider uptake of ROS

The ROS community has expanded gradually over the years as ROS has made its way into a variety of research laboratories, robots and use cases. The wider uptake of ROS and Willow Garage's strategic decisions initiated the transfer of ROS stewardship. In 2012, the Open Source Robotics Foundation (OSRF), a non-profit, was established with a mission *"to support the development, distribution, and adoption of open source software for use in robotics research, education, and product development"*. Around the same time, ROS-Industrial (ROS-I) was started in order to bring ROS and the skills and capabilities of the ROS community into manufacturing environments. The first ROSCon, a ROS developer conference, saw daylight also in 2012. These developments are described and discussed below before presenting ROS in numbers.

While ROS was growing in popularity, much of development and coordination remained in the hands of Willow Garage. Although Willow Garage's effort was of central importance in the development of ROS and the surrounding community, its central role and perceived control of source code made other companies and organisations wary of investing and contributing to ROS development. Therefore, Willow Garage started seeking an opportunity and funding to establish ROS as a self-standing entity outside Willow Garage.

The Defense Advanced Research Projects Agency's (DARPA) Robotics Challenge (DRC) offered a fitting opportunity. Subsequently, OSRF was founded in 2012, and it received a two and half year's contract from DARPA to provide a simulation environment for the Robotics Challenge. The first round of the challenge was run in a simulated environment before the field trials and finals, which were carried out with actual robots in a physical environment. This way the first commission of the foundation did not focus on ROS as such but on the simulation using Gazebo, a robot simulator.

With the newly established foundation, the gradual move of ROS and Gazebo to OSRF began. A team of simulation developers moved to OSRF where they further developed Gazebo and made it accessible as a cloud-based service. This way, at the beginning, OSRF worked on simulation software while the coordination and release management of ROS remained in the hands of Willow Garage. However, as Willow Garage was about to cease its active operations, OSRF was set to take up the responsibility of ROS as well. To this end, OSRF secured funding through National Robotics Initiative (by the National Aeronautics and Space Administration (NASA)), which allowed the foundation to employ ROS developers from Stanford University and Willow Garage among others. With the funding and team in place, the stewardship of ROS and the surrounding technical infrastructure, as well as the open-source community, was transferred to OSRF in 2013.

Although much of the early funding originated from the government contracts and agencies such as DARPA, NASA and the National Science Foundation, the wider uptake of ROS in commercial domains has generated increased funding from for-profit companies. Considering that a non-profit cannot derive a majority of its funding from the for-profit companies according to the financial regulations that govern OSRF, a for-profit subsidiary, the Open Source Robotics Corporation (OSRC), was established in 2016 to manage the commercial funding. To date, the commercial funders include companies such as Google, ARM, Qualcomm, Intel, Canonical, Bosch, Toyota and Mathworks as well as a variety of other companies which make robot hardware platforms, sensors and actuators. At the time of writing, OSRF/OSRC employs some 25 people.

The first steps towards ROS-Industrial (ROS-I) were also taken in 2012. It began as a partnership among Yaskawa Motoman Robotics, Southwest Research Institute and Willow Garage. The mission of ROS-I was to bring the skills and capabilities available in the wider ROS community to the domain of industrial manufacturing. The industrial robotics was seen as a stagnant domain where hardware had reached a higher level of sophistication than software. The industrial grade hardware and lower-level control software were robust and reliable, but the use of proprietary software architectures hindered the reuse of software and replication of research results. Moreover, the aim was to move the

industrial robotics from the era of pre-programmed motion trajectories to the perception based and dynamic motion trajectories. This provided the ROS-I community with a starting point. In the beginning, the people behind the initiative developed hardware drivers and ROS integrations for different robot arms to be able to control them through ROS-based software. Gradually, focus moved from basic integrations to the development of higher-level capabilities, such as perception and motion planning functionalities and calibration tools, as they were seen relevant from the perspective of industrial robotics.

Since the early days, ROS-I has grown in size and global coverage. The ROS-I Americas consortium was established in 2013 by the Southwest Research Institute. This was followed by the consortium in Europe in 2014 by Fraunhofer IPA in Stuttgart, Germany, and in the Asia-Pacific in 2016 by the Advanced Remanufacturing and Technology Center in Singapore. The global consortia bring together a variety of research institutes as well as prominent manufacturers and users of industrial robotics, which facilitates the pooling of knowledge, needs and financial resources to address problems that are pertinent in the field of industrial robotics. Currently, the consortia have some 60 members, including companies such as Caterpillar, John Deere and 3M. The ability to distribute the costs and efforts of software development and maintenance makes advanced industrial robotics more accessible for a wider group of organisations. Also, along with the development and maintenance of domain-specific software, the consortia organise conferences, training and tutorials while collaborating with OSRF.

Furthermore, there are also other industry and domain-specific initiatives, for example around agriculture and defence applications. Also, several academic research programmes build on ROS. Moreover, ROS has made its way to some commercial products, such as Baxter, a collaborative manufacturing robot, from Rethink Robotics and the warehouse logistics robots from Fetch Robotics among others.

The year 2012 also marked the starting point of the yearly ROS developer conference ROSCon. ROSCon brings together ROS users and developers from academia and industry. To date, they have been held as two-day weekend events

in conjunction with some larger robotics and automation related conferences. Whereas the focus of the main conference primarily lies on the research of particular algorithms and computational methods for the purposes of robotics, the focus of ROSCon revolves around software integration and engineering. In other words, instead of individual algorithms, focus is on how to bring and join a variety of algorithms together to build functioning robots. The ROSCon attendance figures have increased over time. Whereas the first ROSCon brought together some 200 attendants, the 2016 conference in Seoul had 450 attendants. The conference is typically run around a single track and consists of both short and long presentations. The presentations cover topics from the fundamentals of ROS, key components and software libraries to their application in different domains, contexts and task-specific use cases. Application areas range from warehouse and humanoid robots, cars and drones to perception pipelines and navigation, to provide a few examples. The acceptance rate of presentations is around 30%. Also, to make conferences available to those who are not able to join at the location, the presentations are streamed live, and recordings are made available online after the conference. In addition, smaller local and regional ROS workshops are also organised as community effort.

The ROS wiki and ROS community metrics[12] report offer quantitative information on the diffusion of ROS. In terms of robot hardware, approximately 100 different ROS compatible actuators and hardware platforms are listed in the ROS wiki. In addition, slightly over 100 different sensors have been made compatible with ROS, ranging from range-finders and 3D sensors to cameras, motion capture, pose estimation and force sensors to give a few examples. In terms of software, the number of software packages exceeds two thousand, some of which are large and of central importance whereas others are of lesser importance or cater to small niche use cases. The ROS codebase contains contributions from some 2000 developers.

In July 2016, 113 000 unique users downloaded ROS packages from the servers hosted by OSRF. However, the real numbers might be higher as the number of downloads does not include downloads from the eleven mirrored ROS

[12] wiki.ros.org/Metrics – ROS Community Metrics Reports

repositories, of which three are in Europe, three in America and five in Asia. In the same period, the ROS wiki, which contains some 17 000 pages, had approximately a million unique page views. answers.ros.org, a discussion forum for technical problems, had around 334 000 page views and slightly less than 15 000 registered users. Similarly to software downloads, the numbers do not include traffic from the mirrored sites. While these numbers are not definitive given the permissive open-source licensing and distributed hosting, they indicate the magnitude of adoption and diffusion. In academic circles, the seminal ROS paper has been cited about 3600 times, which is a relatively high number in comparison to other papers that deal with the robot software development.

With reference to the numbers presented above, it can be concluded that the diffusion of ROS is not of the same magnitude than the Linux operating system or smartphone applications in Apple's App Store or Google Play Store. For them, downloads are counted in billions and the number of developers in millions. However, in the field of robots and autonomous systems, ROS plays a significant role bringing together a vibrant community of roboticists from academia and industry.

5.5 From ROS 1 to ROS 2

Regardless of the wide uptake, ROS is not without shortcomings. It is often said that ROS is good for prototyping and building proofs of concepts, yet the production of a final product could not rely on ROS when reliability and performance requirements are high. To address shortcomings, the next-generation ROS is being developed.

Some of the shortcomings can be derived back to the architectural characteristics of the ROS communication system and initial design assumptions. The approach for establishing and coordinating connections between computational processes contains a single point of failure, meaning that a failure in one particular process can render a whole robot system unmanageable. Also, the communication system cannot guarantee the service levels and verifiability required in mission-critical hard real-time applications. Moreover, the security of interprocess communication was not part of the initial

design requirements. In addition, as ROS was developed with reference to the PR2 hardware, some hardware-specific assumptions are embedded in the software architecture, such as the focus on single robots, the reliability of network connectivity and the availability of computational resources. However, many robotics applications run on smaller processors, operate over unreliable network connections or span over multiple robot systems.

Although the community has developed inventive workarounds to address many of the shortcomings for example by developing real-time capabilities and enhancing service levels, adding redundancy to avoid single points of failure and added layers for security, the development of the next generation of ROS was seen necessary to rectify the issues that derive from the underlying design decisions and implementations.

The discussion on the next generation ROS started gathering pace in 2012 and OSRF announced in 2013 its plan to reimplement the communication system that underlies ROS. It was decided that the current custom-made communications framework would be redesigned and reimplemented using existing and tested technologies that support distributed computation. The project was named as ROS 2, and the first alpha prototype was expected to be available in the first half of 2014. Several existing middleware and communication frameworks were evaluated, and the Data Distribution Services (DDS) standard maintained by Object Management Group (OMG) was selected as the foundation of ROS 2. DDS is a well-received and document standard, which is used in commercial and mission-critical applications such as air traffic control, self-driving cars and spacecraft. In addition, there are several commercial and open-source implementations of the standard.

However, even when building upon an open and tested standard, the design challenge remains on how to design and implement appropriate layers of abstraction between the underlying DDS-based communication system and the ROS-based messaging and coordination systems. Regardless of the optimism, dedication and contributions from the wider community, designing and implementing a communications system has proved time-consuming. The first alpha version was released in September 2015, whereas the eighth iteration of

alpha was released a year later. The third beta version was released September 2017 and the current estimate for the first official release, the version 1.0 of ROS 2, is expected to be released by the end of 2017. ROS 1 and ROS 2 are expected to live side by side for some time since transferring the community and numerous software packages and functionalities from ROS 1 to ROS 2 is expected to take several years.

The experience from the developing of ROS2 sheds light on the difficulty of establishing requirements and implementing systems that facilitate the distributed development and management of complex computational processes.

5.6 Summary

This case description shows how ROS has evolved, changed and expanded over the past ten years. The two university-based research projects which began with an intention to bring a variety of technologies from different subfields of artificial intelligence together to build robots for mobile manipulation tasks in home and office environments turned out as a global open-source community.

Building upon the experiences from the PR and STAIR projects, Willow Garage developed the PR2 robot hardware platform and ROS software framework. Leveraging the lessons learned from the Switchyard communications library, ROS provided an approach for resolving organisational and technological challenges which emerge from the distributed development of distributed computation and the distributed knowledge and control of complex technologies. While the company's efforts in commercialising general robot hardware platforms did not materialise, the effort to establish ROS as an open-source community produced a favourable outcome.

Since the early days, ROS has attracted attention from the wider robotics community, including users and contributors from academia and industry. The ROS communication system, development tools and related infrastructure provide foundational technologies and tools for constructing robot systems as sets of interconnected clusters of computational processes, whereas the ROS community and domain-specific initiatives such as ROS-Industrial bring together technologies, knowledge and resources that are spread across

researchers and developers in different organisations. This provides an overall framework that facilitates technological integration even if control and knowledge of complex technologies are highly distributed.

The next chapter takes a closer look into ROS to examine in the view of its subsystems, combinations and their salient characteristics. This is expected to shed light on how the tensions between the specificity of designs and distributedness of knowledge and control can be resolved.

6 Results of analysis

This chapter reports empirical and conceptual findings of this research. They derive from the thematic analysis that produced six themes and 15 categories as described in Chapter 4. These themes and categories are further developed into models to present conceptual answers to the operative and principal research questions. The chapter unfolds in two parts. Part 1 develops a structural-functional model that conceptualises robots and autonomous systems as contextually bound and embodied chains of transformations. Part 2 develops the notion of the generative-integrative mode of development, which outlines how a generative combination of subsystems and components is crafted iteratively to a composition that produces meaningful context-specific behaviour.

Part 1 develops a conceptual answer to the operative research questions. It begins by presenting a summary of the themes and categories derived through the process of thematic analysis. The themes and categories are first summarised in Section 1 (6.1), which is followed by a more detailed exposition in Section 2 (6.2). The relationships among the themes and categories are then conceptualised as contextually bound and embodied chains of transformation in Section 3 (6.3) that concludes Part 1.

Part 2 builds on the conceptualisation proposed in Part 1 and presents the answer to the principal research question. To this end, Section 4 (6.4) describes and discusses the process and characteristics of software development in the context of complex digitised products and presents a conceptualisation that characterises the unfolding and dynamics of combination in the development of contextually bound and embodied chains of transformation. Subsequently, Section 5 (6.5) elaborates on the role of under-specification and constructive ambiguity in this process, shedding light on how the tensions between the specificity of designs and the distributedness of knowledge and control can and cannot be resolved.

Part One

Robots and autonomous systems as contextually bound and embodied chains of transformation

6.1 Summary of themes and categories

Thematic analysis revealed six themes that are comprised of 15 high-level categories. During the analysis, the research database was processed in the view of the embedded units of analysis with an aim to identify subsystems, combinations and their respective characteristics. The analysis brought forward a highly heterogeneous and distributed technological and organisational arrangement, consisting of thousands of subsystems, components and combinations. The variety of contextual and behavioural requirements that need to be addressed in the development of robots and autonomous systems renders itself as a variety of physical embodiments and computational models that constitute and control the embodiments. This variety is visibly present in ROS and the surrounding community.

The six themes that emerged from the analysis are (1) robot systems, (2) physical embodiments, (3) communication systems, (4) transformation systems and (5) visualisation and testing systems, and (6) the ROS community and software development. Whereas the first four themes deal with robot systems and their constituent elements, the last two focus more on the software development practices and processes. As described in the section on research design, the findings are presented and reported with reference to the ROSCon presentations, which are listed in Appendices B to G and cited as sources of evidence to provide clear chain of evidence. The sources are cited in the format (Appendix: Code). In addition, they are complemented and supported with documentary evidence, field notes and interviews in order to provide a reader with necessary details and evidence. The themes and categories are briefly introduced below and summarised in Table 5. More detailed exposition follows in the subsequent sections.

The theme of robot systems refers to robot systems which are productive applications or research robots. Then, as robot systems are composed of a variety of subsystems, the subsystems are arranged into three themes that are physical embodiments, communication systems and transformation systems. The theme of physical embodiments consists of sensors, actuators and hardware platforms which couple a robot system with the surrounding environment and

establish the physical affordances and limitations that define what a robot system is capable of achieving in relation to its environments. Subsequently, the themes of communication systems and transformation systems refer to computational models and arrangements which produce and control the behaviour of a robot system. The theme of communication systems refers to the layer of software that coordinates the operation and transfers messages between interconnected computational processes, whereas the theme of transformation systems refers to software components that perform transformations between different types and representations of data, such as transformations between geometric coordinate frames or transformations from sensory inputs to actions. The two categories of transformations are often highly intertwined. In sum, physical embodiments, communications systems and transformation systems in combination constitute the architecture of a robot system.

In shifting the focus outside a robot system, two themes emerge. The theme of visualisation and testing systems refers to the type of software that is used to examine, verify and improve the behaviour of a robot system. This includes visualisation, simulation and data management software. Finally, the theme of the ROS community and software development deals with the digital infrastructures and organisational efforts to facilitate the reuse of software and knowledge transfer in the ROS community.

The first five themes are discussed in Section 2 (6.2), whereas the sixth theme, the theme of ROS community and software development, is discussed in Section 4 (6.4).

Theme	Category	Description	Examples of subsystems	Characteristics of combination
Robot systems (Appendix D)	Productive applications	Robot systems developed to perform productive tasks.	Industrial automation, space exploration, self-driving cars.	Robots systems as products and applications that can be used to perform specific tasks and transferred between similar task environments.
	Research robots	Robot systems developed for research purposes, robot competitions and challenges.	Robots used in research laboratories and/or to take part in challenges and competitions, such as the Amazon Picking Challenge and DARPA Robotics Challenge among others.	Research robots are not readily applicable to different tasks or task environments. The purpose of research robots is to generate new knowledge and technologies.

Table 5: Summary of themes and categories

(Table 5 is spread over six pages as per the themes and related categories)

Theme	Category	Description	Examples of subsystems	Characteristics of combination
Physical embodiments (Appendix G)	Sensors	Hardware components that measure the conditions of the surrounding environment.	Cameras, range-finders, force and motion sensors.	Sensors connect a robot system with the surrounding environment. Sensors are connected to the computational processes through the hardware drivers and communication channels. Some conventions for data formats and the transfer of sensory data exist.
	Actuators	Hardware components that exert forces to cause change in the surrounding environment.	Robotic arms, mobile bases and grippers.	Actuators connect a robot system with the surrounding environment. Actuators and their low-level control software are connected to the higher-level computational processes through the hardware drivers and communication channels. Some conventions for the data formats and movement primitives exist.
	Platforms	Hardware which encapsulates sensors and actuators into a single unit.	Quadcopters, cars, mobile devices, teaching platforms, manipulation units.	Connect a robot system with the surrounding environment. Hardware platforms and their lower-level control software are connected with the higher-level computational processes through the hardware drivers and communication channels.

Theme	Category	Description	Examples of subsystems	Characteristics of combination
Communication systems (Appendix B)	Messaging system	Software systems that perform run-time messaging among computational processes.	Messaging methods such as: Topics: one-way transmission Services: request-response Actions: goal-directed actions Some common message type and format conventions which are open for modification.	The ROS messaging system framework provides messaging paradigms for arranging connections between computational processes. The arrangement depends on the composition of hardware and computational processes. Some conventions exist for message formats and data types, but new ones can be created when needed. The conventions enable the syntactic but does not guarantee the semantic interoperability.
	Coordination system	Software systems that manage, coordinate and monitor the messaging system.	Coordination methods for the start-up and runtime configuration, hardware diagnostics and monitoring of software and communication patterns.	The ROS coordination system framework provides paradigms for arranging the connections to coordinate and support the operation of computational processes. While some conventions exist, the configuration and actual workings depend on the physical embodiments and computational processes.
	Connectors	Software components that connect hardware and software components to ROS.	Communication libraries, hardware drivers and bridges to connect to other software frameworks and web applications and user interfaces.	ROS provide a variety of more or less purpose-specific connectors. They are used to connect the messaging and support systems with hardware components and software components such as software libraries, web applications and user interface applications to operate a robotic system.

Theme	Category	Description	Examples of subsystems	Characteristics of combination
Transformation systems (Appendix E)	Coordinate transformations	Software systems that perform transformations over geometric coordinate frames.	Robot model definition, geometric coordinate frame transformation library, robot pose estimation and calibration tools.	ROS provides frameworks for defining and implementing transformations over the coordinate frames and time. The definition and implementation of a transformation system depends on the composition of the embodiments, sensors and actuators and related computational processes.
	Representation transformations	Software systems and components to that perform transformations over representations of the formats and types of data. (for example, to transform perception to motion or localisation to locomotion).	Common packages fsuch as MoveIt! (motion planning), OctoMap (3D occupancy grid), OpenCV (computer vision) and navigation (global and local path planning), among many others.	The ROS distribution contains software packages and/or connections to other packages which contain a number of implementations of perception, motion planning and navigation algorithms among others. While they provide an extensive collection of building blocks, an application specific arrangement of algorithms into a set of interconnected computational processes depends on the surrounding environment, hardware embodiment and expected behaviour.

Theme	Category	Description	Examples of subsystems	Characteristics of combination
Visualisation and testing systems (Appendix F)	Visualisation	Software for visualising a robot's behaviour, physical composition and sensory inputs.	Rviz visualisation package and Rqt dashboard for creating different displays.	The ROS distribution contains a set of tools for the three-dimensional visualisation of robot models and sensory inputs among others. Rqt framework provides a framework for creating plots, diagrams and other displays of a variety of systems data transferred through the communications system.
	Simulation	Software that is used to test and measure a robot's behaviour in virtual worlds that model physical environments and sensory inputs.	Gazebo and other simulators, Morse, USARSim, Webots	Simulations frameworks provide a variety of approaches for creating virtual task environments which model structures, physical features and sensory data of some particular environmental context. Simulations are used widely in development and testing.
	Data management	Software for recording, storing, processing visualising sensory data and behavioural data.	ROS standard functionality for recording and playing back messages. Test data management and analysis packages and services.	Storing and analysing the data gathered during the operation of a robot system is necessary given the low replicability of the situations that unfold in open environments.

Theme	Category	Description	Examples of subsystems	Characteristics of combination
ROS community and software development (Appendix C)	Infrastructure and tools	Digital infrastructures and tools to support the reuse and development of software.	The public build environment and software distribution infrastructure. Private build and integration environments created by different organisations.	OSRF provides a central facility to compile and distribute software from the repositories that are maintained by the community members. However, many organisations manage their own build environments to control the functional dependencies and to protect the intellectual property.
	Knowledge transfer	Organisational efforts and digital infrastructures to transfer knowledge within the community.	Collaboration venues and efforts such the wiki site, discussion forums, the ROSCon conferences and domain-specific initiatives.	The ROS community is heterogeneous. There are several ongoing initiatives to bring people and knowledge from different technological areas together and to promote the transfer of knowledge and the reuse of software.

6.2 Robot systems and their constituent elements

This section expands the description and discussion of the themes and categories introduced above. The themes of robot systems, physical embodiments, communication systems and transformation system and visualisation and testing systems are each presented in their own subsection.

Each subsection has three elements. The first element outlines the purpose and functionality represented by the theme and its constituent categories. The second element describes the constituent categories in more detail and illustrates them with examples where applicable. The third element summarises the most prominent observations concerning the characteristics of combination.

The themes are presented in the same order as they are presented in Table 5. They are presented in this order to establish a coherent narrative which gradually unfolds the categories and their relationships. Section 3 (6.3) develops a conceptual model (see Figure 36) that brings different themes and categories together and describes how they are related to each other.

6.2.1 Robot systems

The theme of *robot systems* stands for the machines and systems which are created to carry out tasks autonomously with limited human intervention; they are expected to exhibit autonomous goal-directed behaviour while responding to environmental contingencies. Whereas the role of a human operator in the control of a robot system is set to be reduced, robot systems will have to be equipped with mechanisms that allows robot systems to control their behaviour in relation to the surrounding environment. Moreover, as a robot system's behaviour emerges from the direct and ongoing interaction with the surrounding environment, the focus of design and development expands from the human-machine interface to a broader spectrum of environment-machine interactions, to situations where the environment need to be understood broadly, including people and physical and digital environments alike. In this light, a robot system can be viewed as a subsystem in relation to its environment, while, at the same time, a robot system itself is composed of

different subsystems that constitute it. Therefore, it is beneficial to discuss the characteristics of combination of a robot system in two ways, in relation to the surrounding environment and in terms of the constituent subsystems.

The theme robot systems consist of two categories, namely, the categories of research robots and productive applications. While the previous paragraph outlines the overall functional principle of robot systems, thematic analysis revealed that the two categories of robot systems differ from each other in terms of the purpose they serve; they play different roles in innovation and differ in their characteristics of combination. Research robots are created to advance the technology while productive applications are created to serve some particular, usually a productive purpose.

Research robots are test beds that are used to learn and study the robotics-related research problems. These problems often revolve around some specific domain of electro-mechanical engineering and artificial intelligence or focus on the development of more complete and complex models of behaviour.

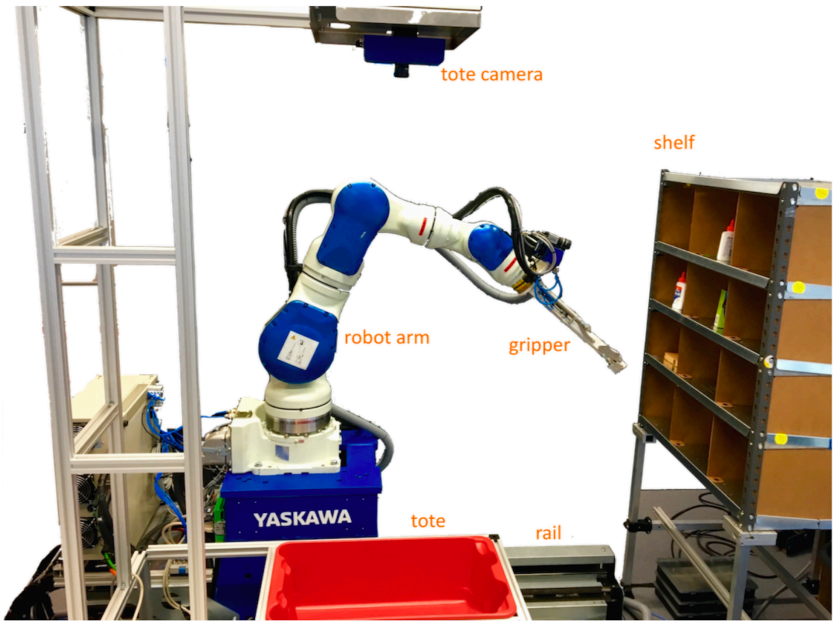

Figure 12: Team Delft robot setup in the Amazon Picking Challenge work cell (Hernandez et al. 2017, D: RS19)[13]

[13] Reprinted by permission from Springer Nature, RoboCup 2016: Robot World Cup XX, Team Delft's Robot Winner of the Amazon Picking Challenge 2016, Carlos Hernandez, Mukunda Bharatheesha, Wilson Ko et al., copyright 2017 Springer Nature.

Problem-specific research projects typically focus on the examination of behaviour of specific algorithms, such as mapping in three-dimensional space (E: TR6), and research outcomes are usually disseminated through the academic journals and software distributions. In contrast, the development of more complete task-specific models of behaviour requires careful integration of a variety of algorithms and computational processes, and robot competitions and challenges provide opportunities to demonstrate skills and technologies in the development of task-specific robot systems. The Amazon Picking Challenge (D: RS19) and the DARPA Robotics Challenge (D: RS15) among the others listed in Appendix D offer examples of such competitions. To illustrate the types of robot systems that are used in competitions, Figure 12 presents the Team Delft's robot that won the Amazon Picking Challenge in 2016 (D: RS19, Hernandez et al. 2017). Figure 13 (D: RS15, Kohlbrecher et al. 2015) presents Team ViGIR's robot that competed in the DARPA Robotics Challenge.

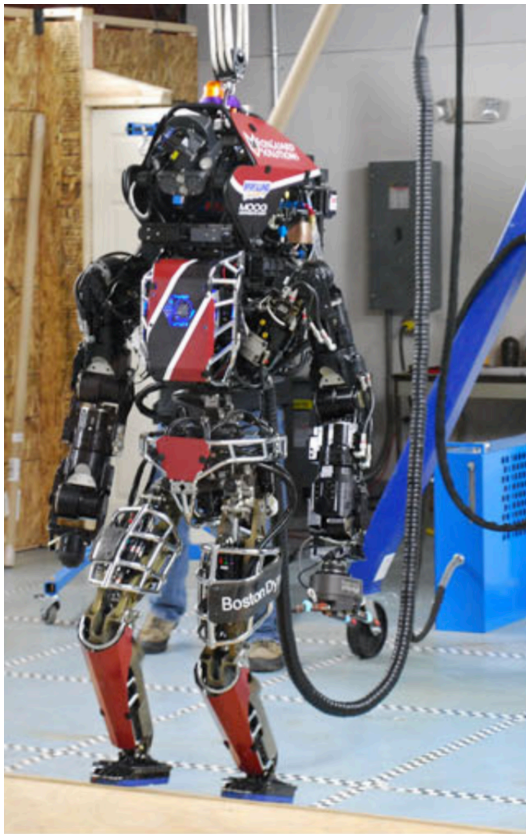

Figure 13: Team ViGIR's Approach to the 2013 DARPA Robotics Challenge Trials (Kohlbrecher et al. 2015, D: RS15)[14]

[14] Reprinted by permission from John Wiley and Sons, Journal of Field Robotics, Human-robot Teaming for Rescue Missions: Team ViGIR's Approach to the 2013 DARPA Robotics Challenge Trials, Stefan Kohlbrecher, Alberto Romay, Alexander Stumpf et al., copyright 2014 Wiley Periodicals, Inc.

Primarily, research robots are developed to explore and experiment novel techniques and technologies and to demonstrate the state of the art in some particular domain of robotics. While some of these projects produce spectacular demonstrations, the robot systems they produce often lack reliability and robustness (D: RS1) that are of paramount importance in the commercial and productive applications. The unpleasant environmental contingencies and the teams of engineers and scientists nursing a robot system are often carefully framed out of a picture when the progress of research is put on a public display.

Productive applications depict a more mundane but equally challenging view of robot systems. Here the success is not defined by demonstrating a novel behaviour in an experimental situation. Instead, practical and commercial success follows from the proposition that the applications that automate operations increase productivity (D: RS21) – and a robot system which needs constant guidance, maintenance and repair is unlikely to do so. Moreover, if a robot system is not robust and reliable, it may harm people and damage the environment it operates in. Therefore, the challenge remains on how to develop robot systems that are productive, robust and dependable with respect to a task and task environment over longer periods of time.

Oftentimes, turning a technological invention to a commercial innovation is a costly and time-consuming process (C: SE16), which requires the exploration of market opportunities as well as the development and fine-tuning of service-propositions and corresponding technological applications. Figures 14 and 15 present service robots, both of which are Willow Garage offspring, that are designed and developed to serve particular market needs. The robot systems from Fetch Robotics serve the needs of warehouse logistics (D: RS18), whereas Relay from Savioke runs indoor deliveries in hotels. Even if these robots are quite similar, performing localisation and navigation on two-dimensional surfaces to transport goods, there are several contextual factors that bear implications to their design and application.

Figure 14: Warehouse logistics robots from Fetch Robotics[15]

Figure 15: Relay, a hotel delivery robot from Savioke[16]

[15] Republished from the website of Fetch Robotics at fetchrobotics.com with permission. Copyright (2018) Fetch Robotics.
[16] Republished from the press kit of Savioke at savioke.com with permission. Copyright (2018) Savioke.

The *characteristics of combination* differ between the two categories of robot systems. Considering that research robot systems are created to serve some particular research objectives, they are rarely readily transferable to different tasks and contexts as such. Instead, the objective with research robots is to study and develop novel technologies that could then be transferred and put to use in other robot systems. For example, algorithms that have been developed, refined and encapsulated into software libraries and packages can be distributed for others to use (E: TR6), something which is one of the underlying objectives of ROS and the ROS community. However, even when software packages are made publicly available, they may not be readily applicable and transferable to other systems and contexts. If a packaged software carries strong assumptions (B: C01) which derive from the context where it was first developed, for example in terms of tasks, task environment or the physical structure and affordances of a physical embodiment, the degree of generality and transferability tends to decrease. Even so, research and engineering projects serve as important sources of methods, tools and technologies that can potentially be further developed, utilised and combined into other robot systems.

The development of productive applications, on the other hand, aims at producing robot systems that carry out productive tasks and are transferable across similar tasks and task environments. While such robot systems can be transferred to an extent, there are physical and behavioural matters which may prevent it. Starting with the physical matters, a robot system has a physical embodiment, sensors and actuators in order to interact with the surrounding environment. For example, the physical embodiment of a pick and place system that works in a warehouse may not be suitable for agricultural fieldwork as the concrete floor changes to soft soil and rectangular boxes to soft berries. Moreover, a behavioural model should produce meaningful actions. To exemplify, whereas the physical embodiment of a self-driving race car might be suitable for transporting people on paved roads, its model of behavioural may not be suitable for normal road transit. Moreover, even if the behaviour of a robot system can be modified by changing software, the computational models of behaviour are bound by the physical embodiment of a robot. Therefore, the characteristics of combination revolve around to what extent a robot system readily combines with the different aspects of tasks and working environments.

To summarise, robot systems are being developed to serve both the research and productive purposes. This differentiation is important in two ways. First, it highlights the different characteristics and objectives of combination, that is, the combinability with respect to constitutive technologies and the combinability with respect to the productive tasks and contexts. Second, it brings forward the differing criteria of success, highlighting the difference between novelty and productivity.

The subsequent section focuses on physical embodiments, taking a more detailed look into their purpose, categories and characteristics of combination.

6.2.2 Physical embodiments

The theme of *physical embodiments* refers to the physical structure and composition of a robot system. The purpose of the physical embodiment is to enable the interaction by coupling a robot system with its physical surroundings. In doing so, a physical embodiment establishes the boundary between the physical environment and the computational model of behaviour. This theme consists of three categories, which are sensors, actuators and platforms. While specialised companies focus on and produce particular categories and types of hardware, purpose-specific embodiments and components are also often developed, such as a custom-made gripper for the Amazon Picking Challenge (Figure 12, D: RS19), depending on the specificity of hardware requirements.

The development of a physical embodiment is subject to a variety of trade-offs (Wyrobek et al. 2008). The distinctive characteristics of tasks and task environments give rise to a number of requirements and constraints, such as required dexterity, payload, speed and safety measures. These requirements then cascade down to the mechanical and electrical design that specify the embodiment of a robot system. During this process, the embodiment becomes specified in terms of its frame and size, the number of joints and their geometric and kinematic relationships as well as the sensors and their capabilities among a number of other factors. These features and properties in conjunction define the underlying physical properties, capabilities and limitations of a robot system.

Sensors measure the environment and supply a robot system with data of the surrounding environment. The sensory information is necessary for the production of contingent behaviour, and the choice of sensors depends on the physical phenomenon and features that are measured to condition and control the behaviour of a robot system. Different sensors harness different physical phenomena, and behavioural models can be conditioned upon a range of measurements, such as the patterns of light and sound, material deformations, the reflections of emitted light and radio waves or rotations and voltages. Each of the measurements is informative in different ways, having particular strengths and weaknesses, modes of failure and idiosyncratic inaccuracies. As the properties of sensors and data tend to converge along the lines of the physical phenomenon that is measured and harnessed, such as two-dimensional images or three-dimensional point clouds (B: CO5), there is a degree of commonality in data formats that can be used to transmit sensory readings between computational processes. Data from multiple sensors are often combined to construct a more comprehensive and reliable representation of the surrounding environment (E: TR3, TR4).

A list of ROS compatible sensors is available on the ROS wiki[17]. Approximately 100 sensors have been made compatible with ROS either by sensor manufacturers or as a community-driven effort (B: CO5). To exemplify, Figure 16 presents a two-dimensional laser range finder from Sick and Figure 17 illustrates Intel's RealSense camera package (G: PE4, PE6). Usually, the details of the ROS integration and interface specification are documented in a software package summary[18][19] on the ROS wiki, whereas the technical details of the capabilities and features of a sensor can be found at the manufacturer's website.

Actuators are the physical embodiments that exert forces in order to cause change. Similarly to sensors, actuators come in many forms and functions depending on their intended use and the phenomena they harness. For example, industrial robot arms (see Figure 18) and grippers embody different degrees of force, motion, dexterity and grasping.

[17] wiki.ros.org/Sensors – A list of ROS compatible sensors.
[18] wiki.ros.org/sick_tim – Package summary for the SICK TiM and the SICK MRS 1000 laser scanners.
[19] wiki.ros.org/RealSense – Package summary for Intel RealSense cameras with ROS.

Figure 16: TiM5xx two-dimensional laser range finder by SICK[20]

Figure 17: RealSense™ D435 camera package by Intel®[21]

Figure 18: Motoman industrial robot arms, MH5F, SDA10, SIA20, by Yaskawa[22]

[20] Republished from the SICK website at sick.com with permission. Copyright (2018) SICK Gmbh.
[21] Republished from the Intel website at intel.com with permission. Copyright (2018) Intel Corporation.
[22] Republished from Yaskawa's Media Center at motoman.com. Copyright (2018) Yaskawa Americas, Inc.

Whereas the heavy-duty robotic arms may exert strong forces to lift and manipulate heavy objects, collaborative robots are expected to be light, flexible and safe enough to share a work environment with people. The robots that share human workspace should be compliant and respond to changes in the external forces in a manner that ensures the safety of workers if they are to collide with a robot system. Similar to robot arms, mobile bases (see Figure 19) are also actuators. They exert forces to make a robot system to move around, and they differ in their characteristics, such as size, payload and the method of steer and drive.

A variety of actuators have been made ROS compatible[23]. This is the case for example with the robot arms[24] in Figure 18 and the mobile base[25] in Figure 19. Again, as different types of actuators tend to converge around their structural characteristics and movement primitives, there is a degree of commonality that facilitates the reuse of the software that commands and controls actuators.

Figure 19: Husky™, a mobile base by Clearpath Robotics[26]

[23] robots.ros.org – A list of ROS compatible robots, actuators and mobile bases.
[24] wiki.ros.org/Industrial/supported_hardware – A list of industrial robot hardware supported by ROS-Industrial (wiki.ros.org/Industrial).
[25] wiki.ros.org/husky_robot – Package summary for Clearpath Husky robot.
[26] Republished from the Clearpath Robotics website at clearpath.com with permission. Copyright (2018) Clearpath Robotics Inc.

Hardware platforms can be viewed as combinations of sensors, actuators and support systems such the body frame and power system, although the boundary between an actuator and hardware platform is not always very clear. Hardware platforms also come in a variety of forms, such as small humanoids (Figure 20, D: RS2, wiki[27]), quadrotors (Figure 21, G: PE8, wiki[28]) or warehouse trolleys (Figure 14, D: RS18). Furthermore, other products and digital devices, such as cars (D: RS16) and mobile devices (B: CO11), can be used as hardware platforms. Mobile devices are rich in sensors and the screen and speakers can be viewed as actuators. To this end, Android-based devices have been made compatible with ROS (B: CO2, CO11, CO24, wiki[29]). As an increasing number of hardware platforms is commercially available, the focus of robot development has been shifting from hardware towards software in some domains of robotics. Making a hardware platform compatible with ROS requires the specification and integration of various physical properties and functional features (G: PE2).

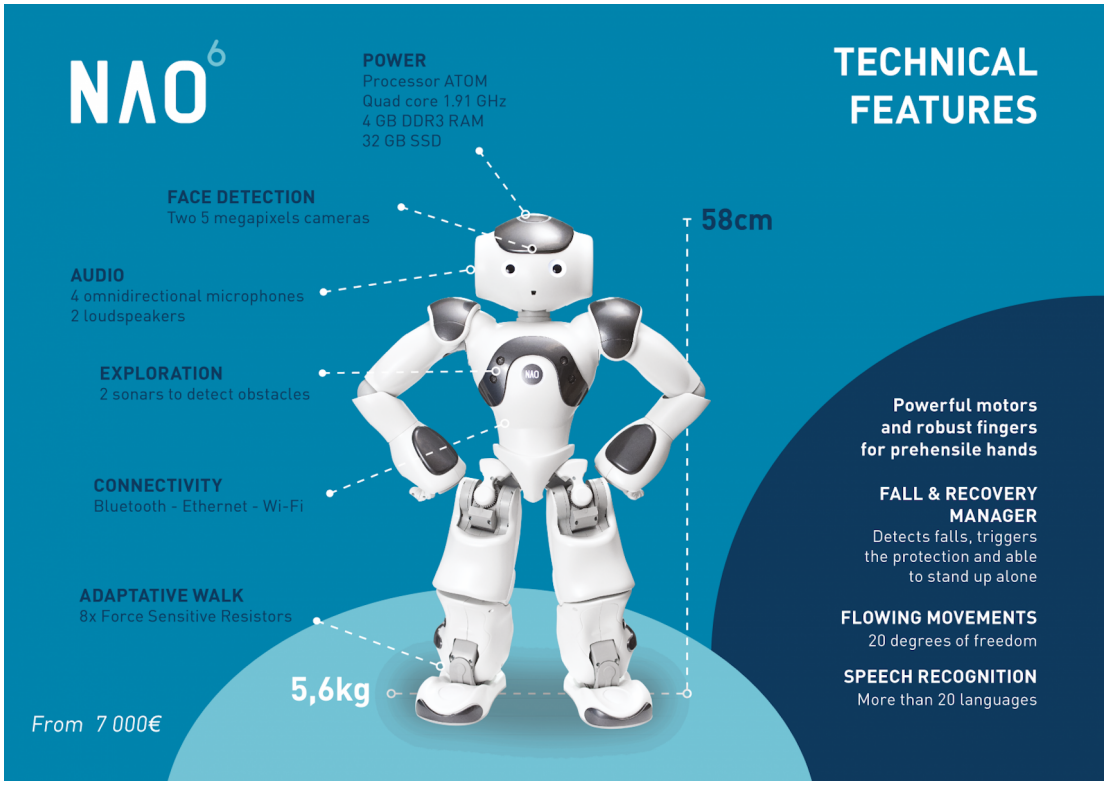

Figure 20: NAO, a bipedal robot by Softbank Robotics[30]

[27] http://wiki.ros.org/nao – Package summary for NAO robot.
[28] wiki.ros.org/dji_sdk – Package summary for DJI onboard SDK.
[29] wiki.ros.org/android, wiki.ros.org/android_ndk – Package summaries for Android integration.
[30] Republished from the Softbank Robotics website at softbankrobotics.com with permission. Copyright (2018) Softbank Robotics.

Figure 21: DJI Matrice 100, a quadrotor for developers by DJI[31]

The *characteristics of combination* concerning the physical embodiments that couple robot systems with their surrounding environments can be approached from the points of view of behaviour and software development. To begin, the physical embodiment of a robot system forms a boundary between the physical and digital, mediating the interaction between the physical environment and the computationally produced behavioural models. The physical world in which a robot system behaves is one of matter, motion and continuity, whereas the digital that produces the contingent behaviour is symbolic, computational and discrete. Combining and aligning the two is not without challenges (C: SE4). Sensors should measure the matters that are computationally tractable so that the behaviour of a robot system could be conditioned upon them, and, at the same time, the forms and functions of actuators and hardware platforms should comply with the requirements of tasks and contexts while also being controllable through computable means. To fulfil these requirements that emerge from environments and operations, a range of different hardware are put to use in the field of robotics to provide robot systems with appropriate embodiments.

[31] Republished from the DJI website at dji.com. Copyright (2018) DJI.

More than 200 hardware components, sensors, actuators and platforms are currently compatible with ROS. The members of the ROS community have developed hardware drivers for connecting the hardware-specific functionalities and firmware with the messaging and support channels of the ROS communication system (B: CO5). Using these drivers that are readily available and distributed as software packages, a range of hardware components can be incorporated into a ROS-based system with relative ease. About a half of ROS compatible hardware are sensors while the rest consists of actuators and hardware platforms.

However, adding or changing hardware is not a simple plug and play operation. As discussed above, different sensors and actuators harness different physical phenomena and differ in their capacities with respect to the physical environment (C: SE4). Moreover, the types and formats of sensory readings, control commands and parameters as well as diagnostics data differ among hardware components and are not uniform nor strictly standardised. There are some commonalities around the interactional modalities, capabilities and physical properties, and this has given rise to some common conventions regarding the formats and types of data that are used when messaging with certain kinds and types of hardware (B: CO1). However, regardless of the conventions that facilitate the integration, developers are expected to remain cognizant of the underlying capabilities and limitations of the physical embodiments as well as their influence on the computational control of the behaviour of a robot system.

The subsequent section focuses on the communication systems and describes their purpose, categories and characteristics of combination.

6.2.3 Communication systems

The theme *communication system* forms a foundational layer for setting up a robot system as a distributed cluster of computational processes. Whereas the physical embodiment couples a robot system with its environment, robot software couples sensors with actuators, mediating and transforming sensory inputs into meaningful actions. This requires an orderly integration of various computational processes, and the communication system plays an essential role

in this endeavour. The communication system consists of three categories, which are the messaging system, coordination systems and connectors.

ROS is designed to facilitate the distributed development of distributed computation for robots and autonomous systems. Its basic architecture can be viewed as a network of interconnected computational processes (algorithms), where processes are interconnected through communication channels which send and receive messages in some agreed format and according to a certain communication paradigm. As illustrated in Figure 6 in the previous chapter, the basic architecture of the communication system can be conceptualised and visualised as a graph (network). The ROS graph defines the composition of a robot's software system in terms of its computational processes (nodes) and their interactions (Quigley et al. 2009). While ROS provides a set of tools and methods to develop and operate robot software as distributed clusters of computational processes, it does not impose any standards on how different computational processes should be partitioned, arranged or how they should interact with each other (B: CO1, CO10). These are design choices which are contingent on the physical embodiment, task and context (as well as to an extent on the contents of reusable software packages). Developing ROS-based software can be seen simultaneously as a process that separates, distributes and combines computational processes (C: SE11).

The *messaging system* carries out operational run-time messaging among distributed computational processes. For this interprocess messaging, ROS provides three different methods of communication: topics,[32] services[33] and actions[34]. First, topics are an asynchronous and unidirectional method of communication. A sending node transmits some specific message constantly at a certain frequency to all nodes that have subscribed to that particular topic. Using topics, sensors can transmit sensory data and measurements as a constant stream. Second, services provision a bidirectional and synchronous

[32] wiki.ros.org/Topics – Topics are named buses over which nodes exchange messages. Topics have anonymous publish/subscribe semantics, which decouples the production of information from its consumption. In general, nodes are not aware of who they are communicating with.
[33] wiki.ros.org/Services – Request / reply between modes is done via a Service, which is defined by a pair of messages: one for the request and one for the reply.
[34] wiki.ros.org/actionlib – The actionlib provides a standardized interface for interfacing with preemptable tasks. Examples of this include moving the base to a target location, performing a laser scan and returning the resulting point cloud and so forth.

method of communication, which is suitable for individual and discrete interactions between two nodes. A node sends out a service request to another node to perform a service and return a response. Third, actions provide a method to handle goal-directed action commands which unfold over longer periods of time, such as a command to grasp an object on the table or to move to some specific location or position. Actions are formed of a set of unidirectional topic streams, and they provide feedback of the status, progress and eventual completion of an action while facilitating the dynamic change of goals and cancellations of tasks. These three standard methods of communication provide software developers with the basic paradigms for setting up interactions among computational processes. Figure 22 (Quigley et al. 2009) provides another illustration of a network of computational processes.

The methods of communication themselves do not define the format or content of transmitted messages. This is done using message[35] and service[36] types that define a format in which messages are transmitted and stored. Depending on the purpose of an interaction, message types may deal for example with stereo and depth images, maps, path plans, waypoints for a global positioning system, coordinates, velocities, commands to actuators or some other intermediate representations and so forth (Quigley et al. 2007). Approximately 75 common[37] message types exist for conventional messaging scenarios, such as sending raw sensor data or dealing with geometry and navigation. Figure 23 illustrates a common and simple message type that expresses a position and orientation on a two-dimensional manifold. If none of the existing message types is able to cater to the messaging needs, new ones can be created by combining primitive data types, such as strings, integers or floating-point numbers of different lengths. These primitive data types have no semantic meaning and they referred to as standard messages. The sending and receiving processes must conform to the same method and message type in order to ensure syntactic interoperability. The configuration of a messaging system can take a number of forms depending on the desired separation and combination of computational processes.

[35] wiki.ros.org/msg – Message description language for describing the data values (aka messages) that ROS nodes publish.
[36] wiki.ros.org/srv – Builds directly upon the ROS msg format to enable request/response communication between nodes.
[37] wiki.ros.org/common_msgs – Common_msgs contains message types that are widely used by ROS packages.

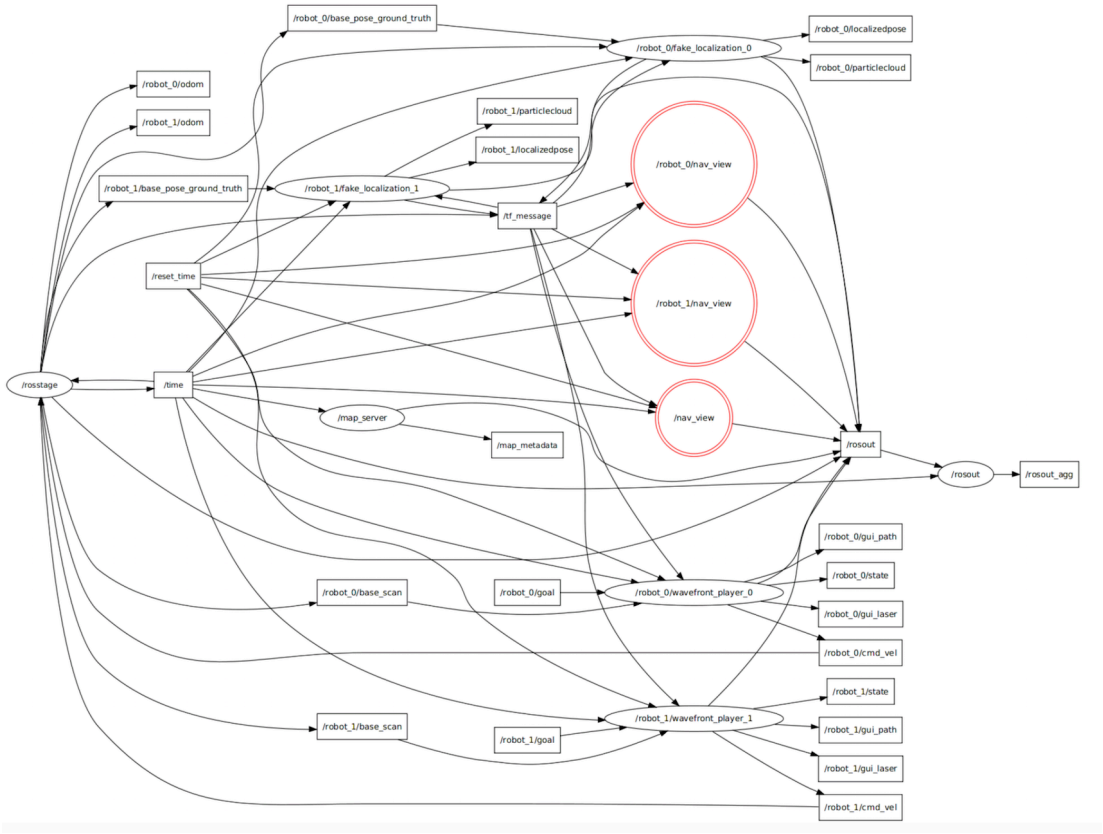

Figure 22: A ROS graph (Quigley et al. 2009)[38]

geometry_msgs/Pose2D Message

File: geometry_msgs/Pose2D.msg

Raw Message Definition

```
# This expresses a position and orientation on a 2D manifold.

float64 x
float64 y
float64 theta
```

Compact Message Definition

```
float64 x
float64 y
float64 theta
```

Figure 23: Pose2D Message that expresses a position and orientation on a two-dimensional surface[39]

[38] Reprinted from ROS: an open-source Robot Operating System by Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler and Andrew Ng., ICRA 2009 Workshop on Open Source Software, 3/3.2, with permission.
[39] Republished from the ROS wiki at docs.ros.org. Creative Commons Attribution 3.0.

Coordination systems manage, coordinate and monitor the operation of a messaging system and computational processes; they handle the meta-data of a robot system and are used to manage and adjust system configuration parameters as well as to monitor the operational state and health of various hardware and software components. To this end, ROS provides a set of basic tools and methods. To begin, the start-up files with roslaunch[40] and parameter server[41] define and initiate a cluster of nodes, their interconnections and other parameters that constitute the run-time configuration of a robot system.

Figure 24 presents a very simple example of a file that launches two nodes, Talker and Listener. The program publisher.py renders the process for the Talker node and subscriber.py for the Listener node, and the programs are located in the package Tutorials. Unlike this simple example, start-up files are often more complex. An advanced robot system may have over hundred nodes running in parallel, interacting in an intricate manner and according to various process and node specific parameters (that are excluded from this example).

```
<launch>
    <node
       name = "Talker"         - Name under which the node operates in the ROS graph
       pkg = "Tutorials"       - Package containing the program that renders the node
       type = "publisher.py"   - Exact name of the program within the package
    />
    <node
       name = "Listener"
       pkg = "Tutorials"
       type = "subscriber.py"
     />
</launch>
```

Figure 24: A simple example of a ROS launch file[42]

The names and types of messages nodes send and receive are specified in the programs that render them. As both sending and receiving nodes must agree on the type and name of a message, ROS enables the dynamic changing of message names for remapping connections without changing the underlying programs as long as the message types that defines data format and structure are congruent.

---

[40] wiki.ros.org/roslaunch – Roslaunch is a tool for launching multiple ROS nodes based on a launch file.
[41] wiki.ros.org/Parameter%20Server – Nodes use the parameter server to store and retrieve parameters at runtime. As it is not designed for high-performance, it is best used for static, non-binary data such as configuration parameters.

In addition, other coordination systems are used to change system configuration parameters at run-time (dynamic reconfigure[43]) and collect and aggregate hardware diagnostics data (diagnostics[44]) (B: CO5). Similarly to the messaging system, there are conventions regarding the diagnostics messages[45], yet new message types can be created if and when needed. The implementation of coordination systems varies from a robot system to another depending on the combination of the physical embodiment, computational processes and messaging system that constitute a robot system. Figure 25 illustrates data elements included in a standard diagnostics message.

diagnostic_msgs/DiagnosticStatus Message

File: diagnostic_msgs/DiagnosticStatus.msg

Raw Message Definition

```
# This message holds the status of an individual component of the robot.
#

# Possible levels of operations
byte OK=0
byte WARN=1
byte ERROR=2
byte STALE=3

byte level # level of operation enumerated above
string name # a description of the test/component reporting
string message # a description of the status
string hardware_id # a hardware unique string
KeyValue[] values # an array of values associated with the status
```

Compact Message Definition

```
byte OK=0
byte WARN=1
byte ERROR=2
byte STALE=3
byte level
string name
string message
string hardware_id
diagnostic_msgs/KeyValue[] values
```

Figure 25: DiagnosticsStatus message for reporting the status of an individual component of the robot[46]

[42] Adapted from a ROS Python and launch file tutorial examples at wiki.ros.org
[43] wiki.ros.org/dynamic_reconfigure – The parameter server provides a means to change node parameters at any time without having to restart the node.
[44] wiki.ros.org/diagnostics – The diagnostics system is designed to collect information from hardware drivers and robot hardware to users and operators for analysis, troubleshooting, and logging.
[45] wiki.ros.org/diagnostic_msgs – Diagnostic messages which provide the standardized interface for the diagnostic and runtime monitoring systems in ROS.
[46] Republished from the ROS wiki at docs.ros.org. Creative Commons Attribution 3.0.

Connectors are used to make different hardware and software components and subsystems compatible with ROS so that they can be incorporated into the ROS messaging and coordination system as nodes. In brief, connectors are software libraries[47] that can be used to envelop another piece of software so that it can establish messaging and coordination channels with other nodes in the ROS graph. Therefore, connectors are an indispensable part of the ROS communication system. Given the variety of computing hardware and programming environments, there are different types of connectors that can be used with different programming languages (e.g. Python, C++, Java, Matlab), microprocessor and microcontroller architectures (e.g. x86 and ARM) and operating systems, such as Linux Ubuntu (C: SE21) and Android (B: CO2, CO3, CO20). Different connectors and methods serve different scenarios ranging from functional software modules to hardware drivers (B: CO15, CO21), user interfaces (B: CO27), application programming interfaces and web-services (B: CO6, CO12) to name a few. Figure 26 illustrates the spectrum of connectors in terms of computing platforms and integration approaches.

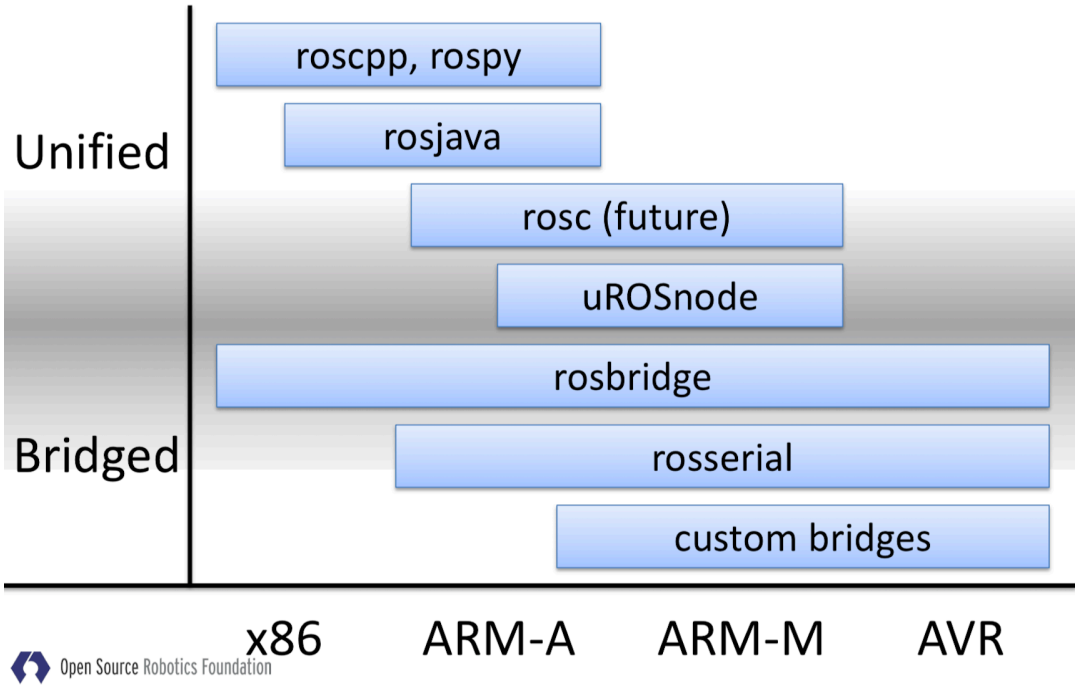

Figure 26: A categorisation of connectors in terms of computing platforms and integration approaches (B: CO15)[48]

[47] wiki.ros.org/Client%20Libraries – A ROS client library is a collection of code that makes many of the ROS concepts accessible via code.
[48] Republished from the 2013 ROSCon presentation Bridging ROS to Embedded Systems: A Survey by Morgan Quigley with permission. Copyright 2013 Open Source Robotics Foundation.

The *characteristics of combination* in the view of the communication systems highlight that computational processes can be distributed and combined in multiple ways at different levels of messaging and coordination. To this end, ROS provides methods, tools, software packages and libraries for setting up the software of a robot system as a distributed cluster of computational processes. Yet, while it provides an overall framework, it remains silent on how the network of computations, messaging and coordination should be arranged; the actual arrangement and integration of computational processes of a particular robot system remain contingent upon the requirements and design choices made during the development of a robot system.

The subsequent section focuses on the computational processes in the view of transformation systems and describes their purpose, categories and characteristics of combination.

6.2.4 Transformation systems

The theme of *transformation systems* characterises much of information processing that is carried out in a robot system. A robot system continuously transforms sensory measurements into goal-directed behaviours in relation to the surrounding environment. Also, the physical embodiment of a robot system consists of parts which constantly move in relation to each other. The behaviour and motion set robot systems in stark contrast when compared to communication systems. Whereas the purpose of a communication system is to transfer data without distortions or delays, the purpose of a transformation system is to convert the inputs of one kind to the outputs of a different kind. In robotics, transformations typically deal with spatiality, perception, decision-making and motion in three-dimensional spaces. This expands the focus on the processes that perform transformations on a variety of representations and types of data.

The theme of transformation systems consists of two categories, which are named here as coordinate and representational transformations. The category of coordinate transformations concerns with the transformations between geometric coordinate frames, whereas the representational transformations are about the transformations between different representations of data in a more

general sense. As the two types of transformations are typically highly intertwined, this separation is analytical and not always readily observable.

Coordinate transformations are about the alignment and transformations between different intrinsic and extrinsic geometric coordinate frames. Of the coordinate frames, the intrinsic refers to the coordinate systems within the embodiment of a robot system, whereas the extrinsic refers to the relationship between a robot system and its environment (E: TR3, TR8). These are not static relationships, and the continuity of motion and behaviour requires frequent transformations between a number of intrinsic and extrinsic frames (E: TR2, TR5, TR13). For example, for a robot system to compute the position and orientation of its gripper in a three-dimensional space in relation to its base at a given moment, time-stamped readings from relevant joint state encoders (sensors) need to be processed with reference to the structural characteristics of the embodiment (E: TR3, TR8). Figure 27 illustrates the kinematic structure of the NAO robot by Softbank Robotics. Similarly, the meaning and fusion of sensory data are predicated on the frame and time of reference. For example, when patterns recognised from camera images are combined and processed to condition the behaviour of a robot system, the accurate framing in terms of space and time is of paramount importance (E: TR3, TR8).

ROS provides general methods and tools for setting up the systems of coordinate transformations. These revolve around the robot model definition (urdf[49]), transformations library (tf[50]) and some common conventions[51][52]. The construction of a coordinate transformation system begins by defining the structural elements and characteristics of the physical embodiment of a robot system, including the core frame and body, joints and links along with their morphologic, geometric, kinematic and visual properties (E: TR1, TR9, TR14). These are specified in the Universal Robot Description Format, in an XML-file that is called urdf for short. Depending on the scope and level of details, these files grow in size and complexity, and the ROS community has developed tools to automate the process of model creation (E: TR9, TR14).

---

[49] wiki.ros.org/urdf – Package summary for Unified Robot Description Format (URDF), which is an XML format for representing a robot model.
[50] Wiki.ros.org/tf2 – Package summary for tf2 transform library.
[51] ros.org/reps/rep-0105.html – REP 105 Coordinate Frames for Mobile Platforms.
[52] ros.org/reps/rep-0120.html – Coordinate Frames for Humanoid Robots.

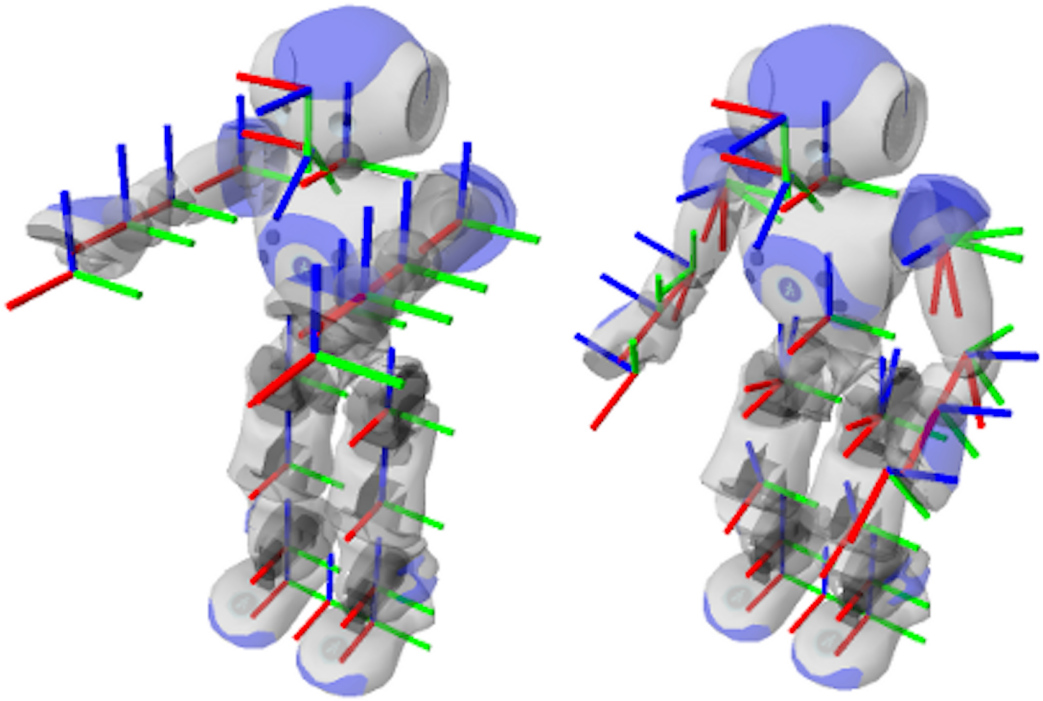

Figure 27: The kinematic structure of a NAO robot[53]

For example, a detailed model[54] for the NAO robot pictured in Figure 27 may include some 2500 lines of definitions. Therefore, urdf is one of the central configuration files as it captures the properties of a physical embodiment and its articulation making it available for different computational processes. In doing so, it provides the parameters of a structural configuration for example to the processes that perform coordinate transformations (E: TR1), localisation and pose estimation (E: TR1, TR16), motion planning, collision checking (E: TR5), robot visualisation and simulation among others.

Building upon the robot model definition, the transform (tf) library (Foote 2013) provides tools and methods for embedding and performing coordinate transformations within computational processes. In addition, some other transformation software libraries have also been made compatible with ROS, such as the Kinematics and Dynamics Library (KDL[55]) form the Orocos project

[53] Republished from the ROS wiki at wiki.ros.org. Creative Commons Attribution 3.0.
[54] wiki.ros.org/nao_description – Description of the Nao robot model that can be used with robot_state_publisher to display the robot's state of joint angles. github.com/ros-naoqi/nao_robot/blob/master/nao_description/urdf/naoV50_generated_urdf/nao.urdf. Creative Commons Attribution 3.0.
[55] wiki.ros.org/orocos_kdl – Package summary for the Kinematics and Dynamics Library (KDL), distributed by the Orocos Project.

and the Bullet[56] physics library, among others. Furthermore, there are tools and methods to carry out intrinsic and extrinsic calibration to ensure the alignment of coordinate frames among the different parts of a robot system and in relation to its environment (E: TR4, TR12, TR17).

Representational transformations are about the processes which transform data from some format or representation to another, and although they are highly intertwined with coordinate transformations, they can be viewed from a slightly different angle. Broadly speaking, they can be viewed as processes that transform sensory inputs into actions. This can be illustrated with a simple example of a transformation pipeline (B: CO1) that performs an action based on a particular visual input. At the beginning of the pipeline, a digital image is captured by a camera. Then, this image is passed on to a pattern and object recognition algorithm that tags objects in images based on certain features. Subsequently, these tags are passed on and used as an input in the process that triggers a particular task when it receives a certain tag. Then, this task is passed on as a command to the motion and trajectory planner, which then sends a more specific actuation command to the hardware controller that drives motors. In other words, a representation of data undergoes a gradual step-by-step transformation from a sensory input into a corresponding action.

Individual steps of transformation can be distributed over different computational processes in multiple ways, even if the overall mapping between the initial sensory inputs and resulting actions would remain invariable. There are no hard and fast rules on how the overall behaviour of a system should be decomposed among different transformations, nor in which format the data should be passed from one transformation to another (C: SE11). However, developing chains of transformation every time from scratch would be a daunting prospect.

To make software development easier, ROS offers reusable software packages as well as reference implementations and conventions on how to combine them. Currently, approximately 2000 ROS compatible software packages and frameworks are available through the ROS software distribution infrastructure.

[56] wiki.ros.org/bullet – Package summary for 3D Game Multiphysics Library provides state of

These packages offer a number of algorithms and functionality that cater to different needs. For example, OpenCV[57], a computer vision framework, contains some 2500 algorithms for various image processing, machine learning and object and pattern recognition purposes (E: TR4), whereas Point Cloud Library (PCL[58]) provides algorithms for processing point clouds and depth images (E: TR4). OctoMap[59] provides a framework for constructing three-dimensional occupancy grids in order to store and query spatial information (E: TR6). In turn, MoveIt![60] is a motion planning framework which combines a variety of algorithms and functionality for three-dimensional perception, motion planning, collision checking and other motion-related purposes (E: TR2, TR5, TR13). In addition, there are many other software packages, stacks and meta-packages that produce different transformation or processing pipelines to cater some specific purposes, for example, to pre-process raw image[61] or point cloud[62] data into useful inputs for vision algorithms (E: TRO4, TR7). Figure 28 illustrates the central functionalities and components of the MoveIt! package at a high level, and Figure 29 depicts the high-level system architecture[63] in terms of main processes and their interconnections.

While the above presentation of the variety of representational transformations is superficial and cursory at best, and new software packages and libraries are constantly being developed, it, however, sheds light on the heterogeneity and complexity of robotics software that can be found among those 2000 software packages that are available through the ROS infrastructure.

---

the art collision detection, soft body and rigid body dynamic.

[57] opencv.org; wiki.ros.org/opencv3 and wiki.ros.org/vision_opencv – Package summaries for interfacing ROS with OpenCV, a library of programming functions for real time computer vision.

[58] pointclouds.org; wiki.ros.org/pcl – Package summary for the point cloud processing with the Point Cloud Library.

[59] octomap.github.io; wiki.ros.org/octomap_ros – Package summary for octopmap_ros that enables a convenient use of the octomap package in ROS.

[60] moveit.ros.org; wiki.ros.org/moveit – Package summary for Meta package that contains all essential package of MoveIt!

[61] wiki.ros.org/image_pipeline – Package summary for the image_pipeline stack that preprocess raw camera images for vision algorithms: rectified mono/color images, stereo disparity images, and stereo point clouds.

[62] wiki.ros.org/laser_pipeline – Package summery for the laser_pipeline stack to preprocess a scanning laser rangefinder data.

[63] moveit.ros.org/documentation/concepts/ – MoveIt! motion planning and control architecture.

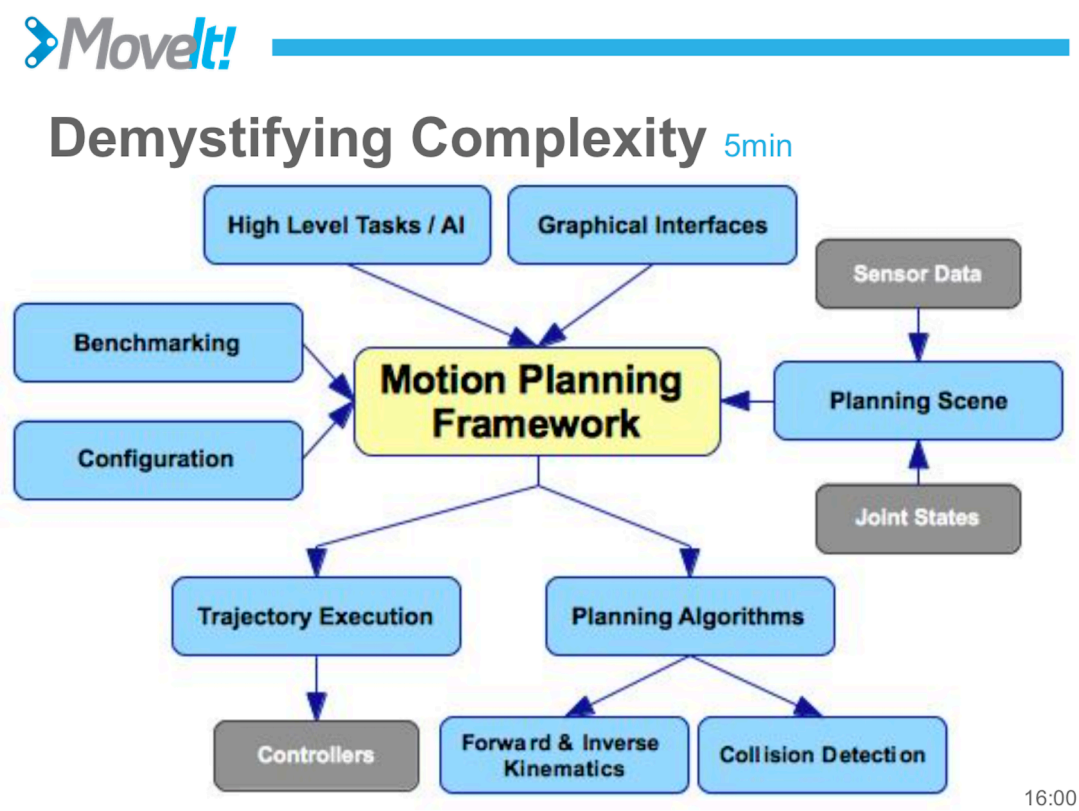

Figure 28: MoveIt! - Demystifying complexity (E: TR13)[64]

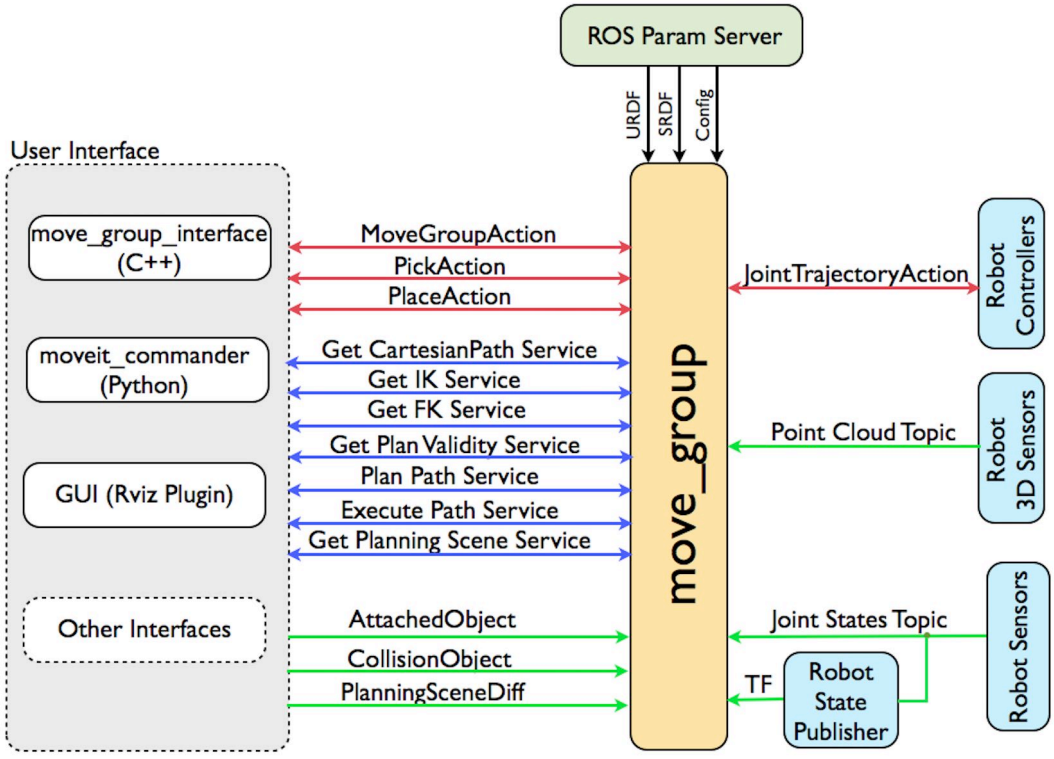

Figure 29: MoveIt! system architecture[65]

[64] Republished from the 2015 ROSCon presentation MoveIt! Strengths, Weaknesses, and Developer Insight by Dave Coleman with permission. Copyright (2015) Dave Coleman.
[65] Republished from the MoveIt! website at moveit.ros.org. Creative Commons Attribution 4.0.

The observation of the *characteristics of combination* in the context of transformation systems brings forward a similar pattern that was observed in the context of communication systems. The software packages distributed through the ROS infrastructure provide frameworks and building blocks, subsystems and components, that can be combined when designing and building a robot system. However, these packages are not ready-made applications that could just simply be installed. Instead, they can often be viewed as end-product agnostic subsystems which can to an extent be combined to produce desired behavioural models for robot systems. In this light, it is not surprising that the possibilities and challenges of joining together different computational processes emerged as a unifying topic and concern.

As transformations grow in size and become more convoluted, they become harder to understand. The next section focuses on visualisation and testing system, which are used to examine and verify the behaviour of a robot system.

6.2.5 Visualisation and testing systems

The theme of *visualisation and testing systems* concerns with the examination and evaluation of robot systems. This is often far from trivial considering the interdependent and intertwined nature of environments, tasks, physical embodiments, communication and transformation systems and the associated interactional complexity that unfolds over time and space. As it is not feasible to simply reason how a robot system would function in a spectrum of situations, developers use a range of digital methods and tools to examine the internal functioning and outward behaviour of a robot system. The theme visualisation and testing systems consist of three partially overlapping categories that cover the systems for visualisation, simulation and data management.

Visualisation systems provide the methods and tools for rendering system configurations, embodiments, sensory data, message contents and motion on the computer screen. In general, the development of a robot system is a highly visual process. While very few of the analysed conference presentations were about visualisation tools as such, they were constantly on display. It appears to be challenging to convey or grasp an idea of any embodied behaviour as a series of multidimensional and parallel movements and motions without relying on

visual renderings. Through visualisation systems, developers can observe the functioning and behaviour of a robot system and examine what causes them to behave in a certain manner.

For visualising, the ROS distribution provides Robot Visualiser (RViz[66]) and the RQT[67] package for connecting ROS with the QT user interface framework. They reflect the heterogeneous and distributed structure of ROS-based robot software by facilitating flexible construction of visualisations that correspond to the structure of a particular robot system.

RQT provides methods and tools to visualise and examine ROS graphs, nodes, messages, coordinate transformations and to plot, diagram and display data transmitted over messaging and coordination channels (F: VI6). Figure 30 provides an example of an RQT dashboard that combines and presents run-time sensor data and system diagnostics. As visualisation tools take their input primarily from communication systems, the resulting visualisations are contingent not only on the features and properties that are of interest to the developers but mirror the architecture of computational arrangements.

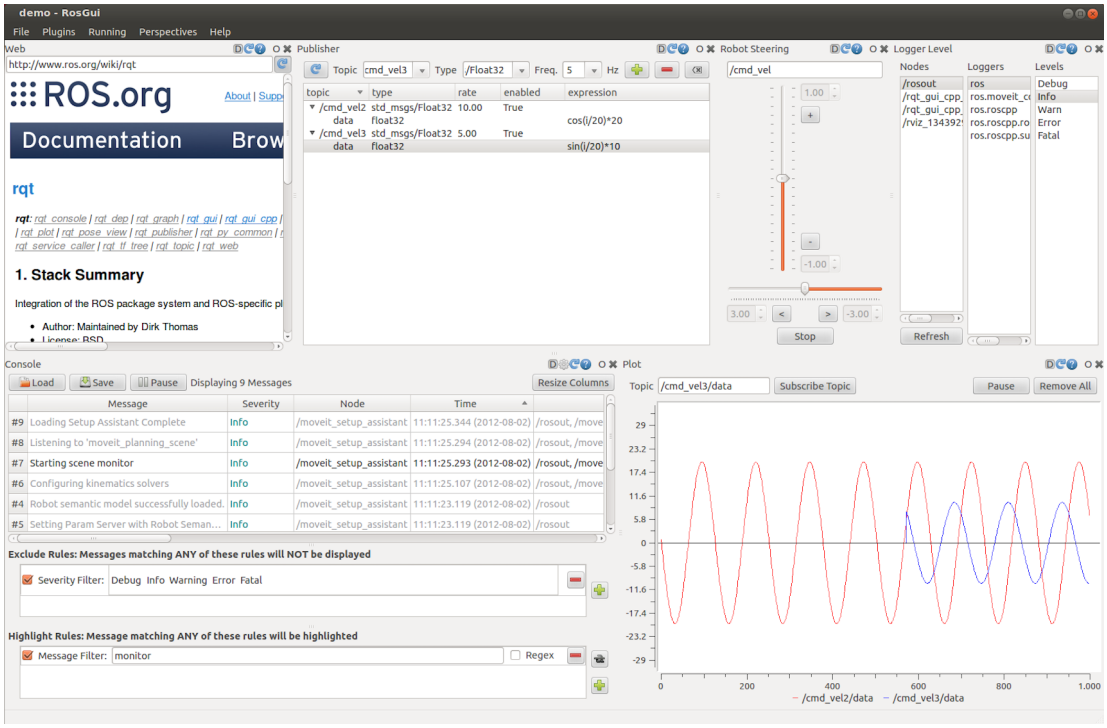

Figure 30: An example of an RQT visualisation dashboard[68]

[66] wiki.ros.org/rviz – Package summary for three-dimensional visualisation tool for ROS.
[67] wiki.ros.org/rqt – Package summary for a Qt-based framework for graphical user interfaces development for ROS.
[68] Republished from the ROS wiki at wiki.ros.org. Creative Commons Attribution 3.0.

RViz provides tools and methods for constructing three-dimensional visualisations by combining robot models (urdf), sensory data, control systems and other contextual information. Overlaying different representations on the screen can be used, for example, to provide a developer with an overview of the embodiment and state of a robot system, its view of the surrounding environment as well as the relation to the other objects, among other things. Developers can also interact with robot systems through the visualisations. Figures 31 and 32 illustrates RViz visualisations of the kinematic structure, motion and sensory data processing of robot systems.

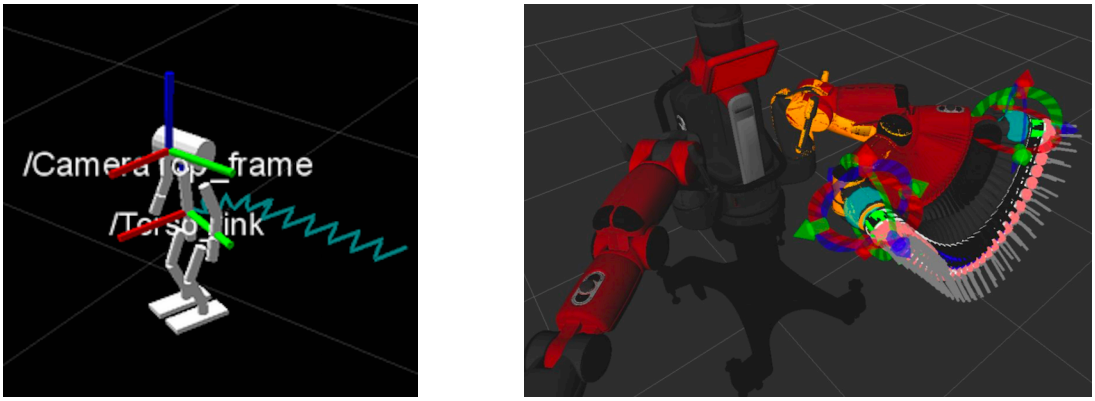

Figure 31: RViz visualisation of the kinematic structure[69] and motion planning of robot systems (E: TR13)[70]

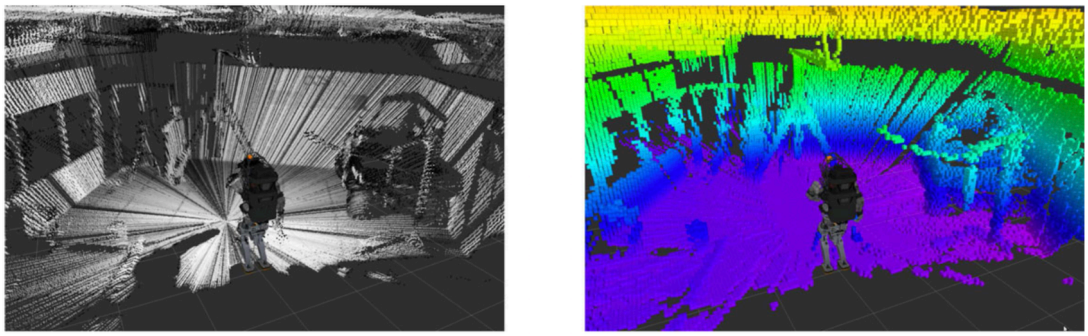

Figure 32: Sensory data processing that shows data from a laser range finger with intensity information and resulting Octomap representation (Kohlbrecher et al. 2015, D: RS15)[71]

Simulation systems are also widely used in the development of robot systems (F: VI1, VI8, VI12). Simulations are used to model and visualise physical

[69] Republished from the ROS news at news.ros.org, Creative Commons Attribution 3.0.
[70] Republished from the 2015 ROSCon presentation MoveIt! Strengths, Weaknesses, and Developer Insight by Dave Coleman with permission. Copyright (2015) Dave Coleman.
[71] Reprinted by permission from John Wiley and Sons, Journal of Field Robotics, Human-robot Teaming for Rescue Missions: Team ViGIR's Approach to the 2013 DARPA Robotics Challenge Trials, Stefan Kohlbrecher, Alberto Romay, Alexander Stumpf et al., copyright 2014 Wiley Periodicals, Inc.

features of target environments, the sensory inputs they provide, the behaviours of robot systems as well as the responses those behaviours may trigger in the target environment. This enables the development and cost-efficient testing of the software of a robot system against a virtual world before moving the code onto a physical robot and actual target environment (see Figure 33). As computational models, simulations also facilitate controllability and repeatability of experimentation, which is difficult to achieve in open-ended physical environments (F: VI14). This way, simulation provides a method to test, analyse and benchmark behavioural models and algorithms under different conditions and in different types of robots (F: VI14). In addition, simulations can be used in continuous integration testing to ensure that a change in some part of a robot system does not change the behaviour over a range of situations in an unplanned manner (E: TR5).

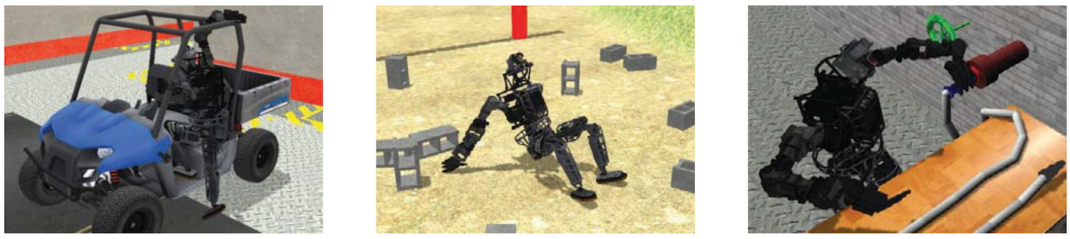

Figure 33: Simulation illustrations from Team ViGIR's approach to Darpa Virtual Robotics Challenge (Kohlbrecher et al. 2013)[72]

OSRF also supports and coordinates the development of the Gazebo[73] open-source simulator (Figure 34, F: VI1, VI8, VI12), and Gazebo is well integrated with ROS. In addition, several other simulators have been made compatible with ROS (F: VI2, VI3, VI9, VI13, VI14). Some of them focus on specific application domains whereas others can be viewed as more general frameworks, each of them having their particular strengths and weaknesses. Figures 34 and 35 illustrate some target environments, use cases and the types of objects and basic elements of which simulation environments can be constructed. Although object models and simulated worlds can be constructed with editors from the ground-up, object libraries facilitate the reuse of simulation objects and models making the building of simulation environments easier.

[72] Reprinted from 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), Overview of team ViGIR's approach to the Virtual Robotics Challenge, S. Kohlbrecher, D. C. Conner, A. Romay, F. Bacim, D. A. Bowman and O. von Stryk. Copyright 2013 IEEE.
[73] gazebosim.org - The homepage of the Gazebo robot simulator.

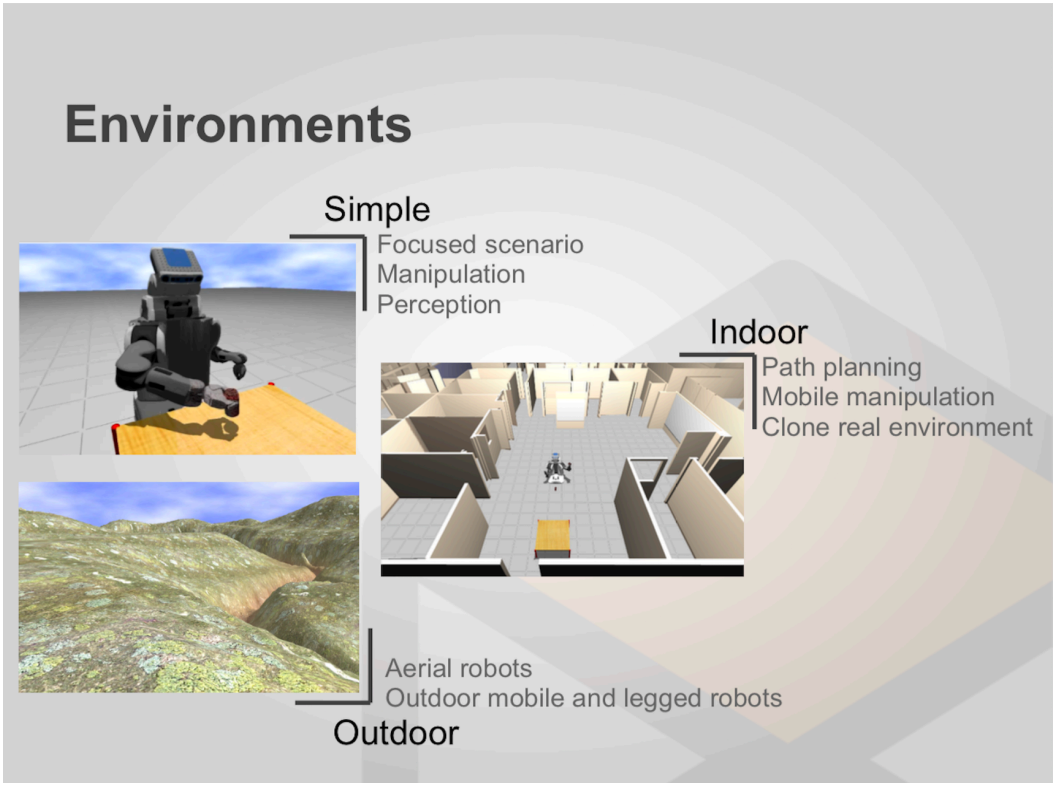

Figure 34: Use cases and target environments for Gazebo simulations (F: VI1)[74]

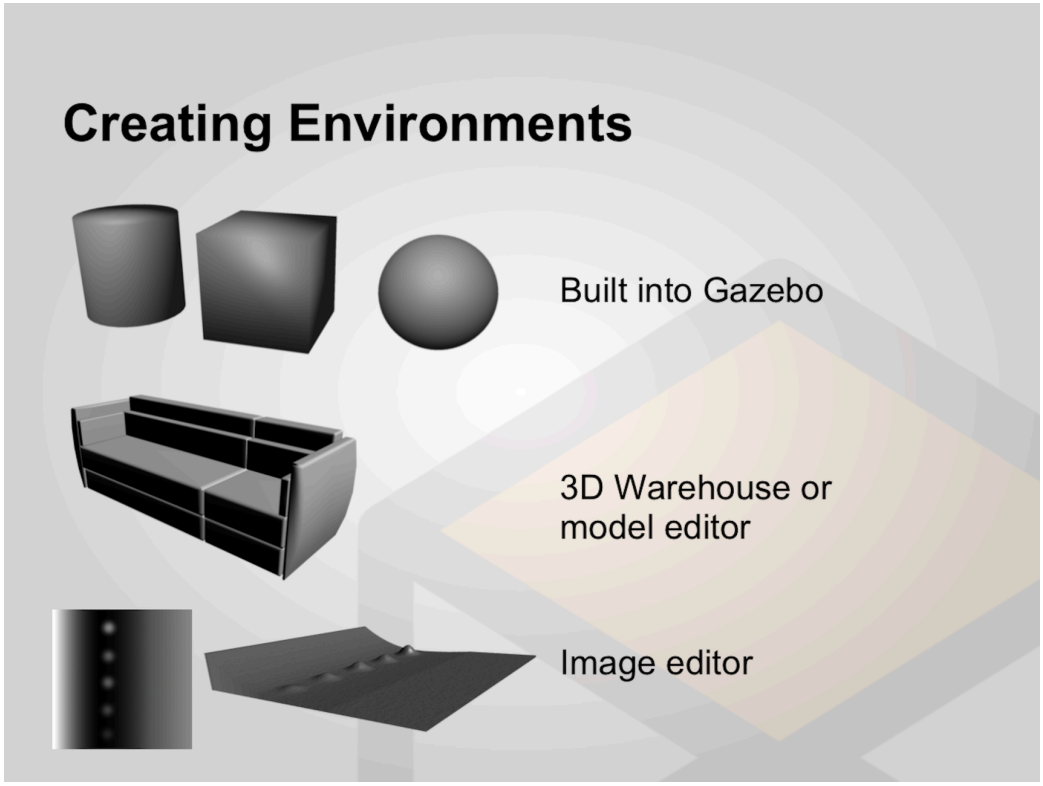

Figure 35: Simulation environments are constructed with editors from different simulation objects (F: VI1)[73]

[74] Republished from the 2012 ROSCon presentation The Gazebo Simulator as a Development Tool in ROS by John Hsu and Nate Koenig available at gazebosim.org. Copyright 2012 Open Source Robotics Foundation.

While simulations are beneficial, running a virtual robot in a virtual environment cannot replace physically grounded testing (F: VI15). Testing in simulation resembles more of a computer game than any real-world scenario.

To reduce the reality gap, robotics companies and research organisations use physical simulation environments where they can run experiments in a physically grounded manner. Figure 36 illustrates the difference by contrasting a simple block model and physical simulation side by side.

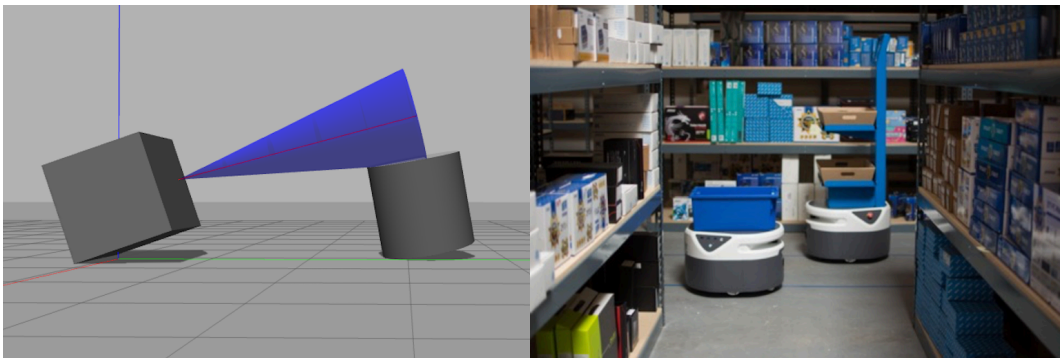

Figure 36: An example that contrasts a simple block model[75] and physical simulation (F: VI15)[76]

Data management systems are used to collect and manage test data. The systematic collection of sensory data and messages transmitted between computing nodes is essential for the purposes of testing and longer-term software development (F: VI7, VI15, VI16). Recording and storing the run-time data enables the replication of scenarios which could not otherwise be reproduced or rerun.

With stored sensory data and messages, developers can rerun scenes and scenarios to develop and test software against rich data sets which represent the unfolding of previous real-life or test situations. In addition, such data sets can further be analysed and annotated so that relevant patterns and features can be extracted to develop algorithms and behaviours which are able to cater to a wider variety of situations (F: VI7, VI16). For the purposes of data collection and

[75] An example of a block model where a sonar sensor cone collides with an object instead of intersecting. Republished from the Gazebo website at answers.gazebosim.org/question/16242/sonar-sensor-cone-has-a-collision/. Creative Commons Attribution Share Alike 3.0.
[76] Republished from the 2016 ROSCon presentation Physical Continuous Integration — CI on Real Robots! by Alex Henning with permission. Copyright (2016) Fetch Robotics.

management, the ROS communication system provides rosbag[77] functionality for recording message streams from different communication channels and to store them into bag files (F: VI7, VI16). These streams can then be filtered, examined and replayed at a later point of time.

Much of data storage, management, analysis and visualisation takes places in digital infrastructures and systems that reside outside a robot system, and there are companies that offer related technologies and services (F: VI16). In addition, there are public repositories where researchers and developers can store and their datasets and make them publicly available. However, the extent to which data can be shared is contingent on the structure of a robot system, the configuration of its communication system, message types and the distribution of computational processes in general. Therefore, while some sensory data recorded with common message types might be readily transferable across different systems, the interprocess data stored using customised and non-conventional message types is often much less so. Similarly, changing message types and computing architecture during the development of a robot system may break the backward compatibility rendering the previously recorded data obsolete (D: RS11).

The *characteristics of combination* concerning the visualisation and testing systems reflect the complexity and heterogeneity of robot systems; visualisation and testing systems are essential in the probing of multidimensional and temporal couplings within a robot system and in relation to its environment. As the multi-layered and continuous transformations are often convoluted and difficult to follow, the purer forms of reasoning and logic appear to give way to observation and experimentation. The characteristics of combination can be reflected through behavioural couplings and technological combinations.

Behavioural couplings can be reflected in terms of the internal functioning and the outward behaviour of a robot system, yet it is worth to note that the two are intrinsically linked as the internal functioning leads to the outward behaviour. To observe and examine various behavioural couplings and relationships, visualisation systems are used to render internal and external representation of

[77] wiki.ros.org/rosbag - A set of tools for recording from and playing back to ROS topics.

a robot system, its behaviour and environment. In addition, simulation systems facilitate the controlled experimentation, whereas data management systems enable the recording, analysis and rerunning of different situations and outcomes. Visualisation and testing tools need to be able to combine elements of a robot system in multiple ways to expose different behavioural couplings. To this end, technological frameworks, visualisation and testing systems provide a variety of methods, tools, technologies and frameworks for developers to combine and construct visual displays and virtual worlds. For example, for simulation purposes, mock-worlds can be constructed from reusable building blocks that mimic target environments and their physical properties with varying degrees of reality. Furthermore, simulation systems offer flexible architectures upon which new simulation worlds, objects and dynamics can be developed. To absorb the heterogeneity and complexity that characterise not only robot systems but also their target environments, visualisation and testing systems manifest themselves even more open-ended, heterogeneous and complex than the robot systems whose behaviour they have been created to observe and evaluate.

The dependence on the visualisation and testing system does not only tell that robot systems are difficult to reason about. It also shows how the methods and tools of system development reflect the expansion of the focus from human-machine interactions to environment-machine interactions. As robot systems interact directly with the environment they are embedded in, developers cannot solely rely on their personal experience through the screen-based interaction as criteria in the evaluation of interaction designs. Instead, they will have to model and construct an environment against which the fitness of a particular behavioural model of a robot system can be examined and evaluated. This demonstrates the shift of focus in the design of interaction and the need for appropriate instruments for observing and making sense of these interactions.

6.3 Robot systems as chains of transformation

This section brings together the themes and categories discussed in the previous sub-sections and develops a conceptual model that illustrates how they are related to each other. The outcome can be viewed as a structural-functional model that conceptualises robots and autonomous systems as *contextually*

bound and embodied chains of transformation. The purpose of this is to make the relationships among the themes and categories explicit so that they can be analysed and theorised further.

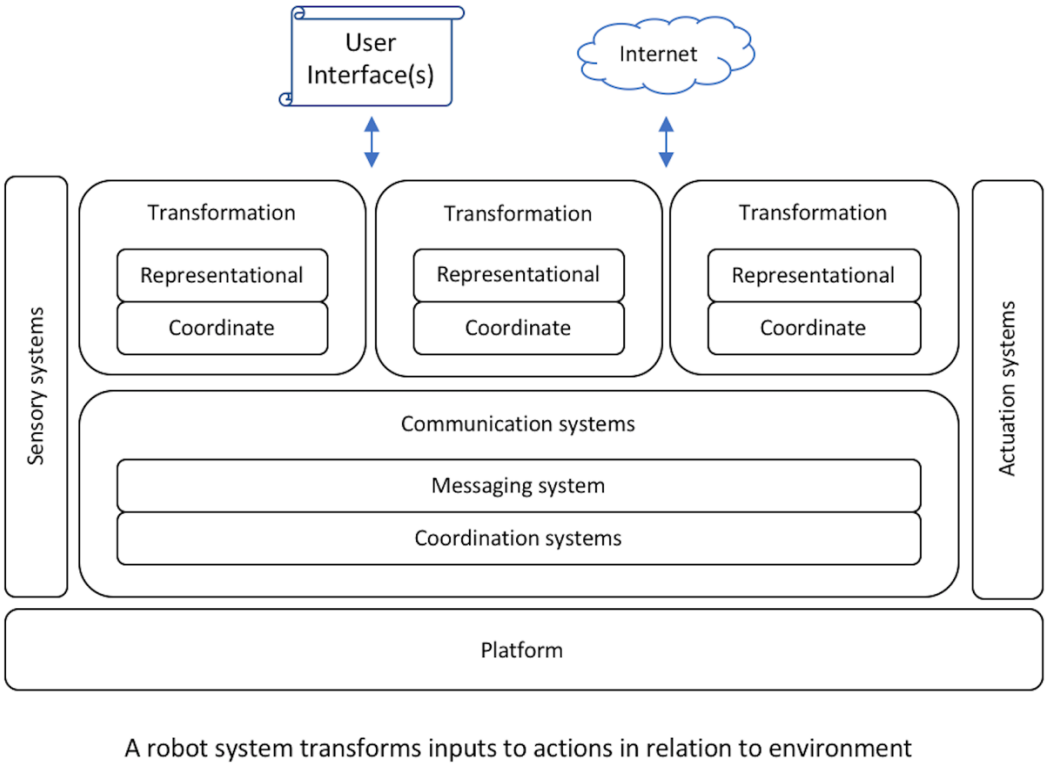

Figure 37: A conceptual model of a robot system (Own figure)

A structural scheme of a robot system can be presented as a composition which comprises three thematic groupings, that is, physical embodiments, communication systems and transformation systems, as presented in Sections 6.2.2, 6.2.3 and 6.2.4.

Figure 37 illustrates how these thematic groupings are related to each other. Here, the conceptual model is sketched out as a stack-like structure, which presents an idealised and simplified composition of a robot system. First, the blocks on the bottom and the sides represent the physical embodiments which define the boundary of a robot system in relation to its environment. Sensors, actuators and hardware platform constitute the embodiment that allows a robot system to interact with the surrounding environment. In doing so, these embodiments mediate between the physical and digital matters. Second, the three blocks on the top of the communication system block represent a variety of transformation systems which perform transformations upon geometric

coordinates and other representations of data, and these transformations define the way in which sensory inputs are transformed into actions. Finally, the middlemost block represents the communication systems which connect the physical embodiments and different transformation systems to each other while also coordinating and monitoring their operations. The communication systems can be viewed as a unifying layer through which the distributed physical embodiments and transformation systems come to interact with each other.

It is worth note that this illustration is limited in a few important ways. The two-dimensional layered stack-like illustration does not expose the distributed and networked computing architecture of robot systems, as illustrated earlier in Figures 6 and 22 when the architectural principles of the ROS communication were presented. Also, the number of transformation systems is typically much higher than three and they may interact with each other in a highly intricate and parallel manner. In addition, sensors, actuators and platforms come in many different forms and configurations that transcend the rigidity of this illustration.

However, the presented conceptualisation illustrates the main thematic groupings while showing how they are related to each other. The communication systems and related methods and tools can be seen as a foundational layer, and it also forms the unifying framework that brings the ROS community together. In turn, the physical embodiments and transformation systems show a greater degree of variety revealing the spectra of forms and functions that reflect the variety of contextual and behavioural requirements and couplings. These three thematic groupings in combination render the structure of a robot system.

To elaborate this conceptualisation further, the structural focus presented above can be complemented with a functional one. For this purpose, it is beneficial to include the contextual embeddedness in the picture and bring forward the role of interactional and behavioural couplings as they play an essential role in the animation of robot systems. Thereby, the robot system and its constituent elements illustrated in Figure 37 are added with the layers that illustrate the contextual embeddedness and interactional characteristics of behavioural models. These additions are presented in Figure 38.

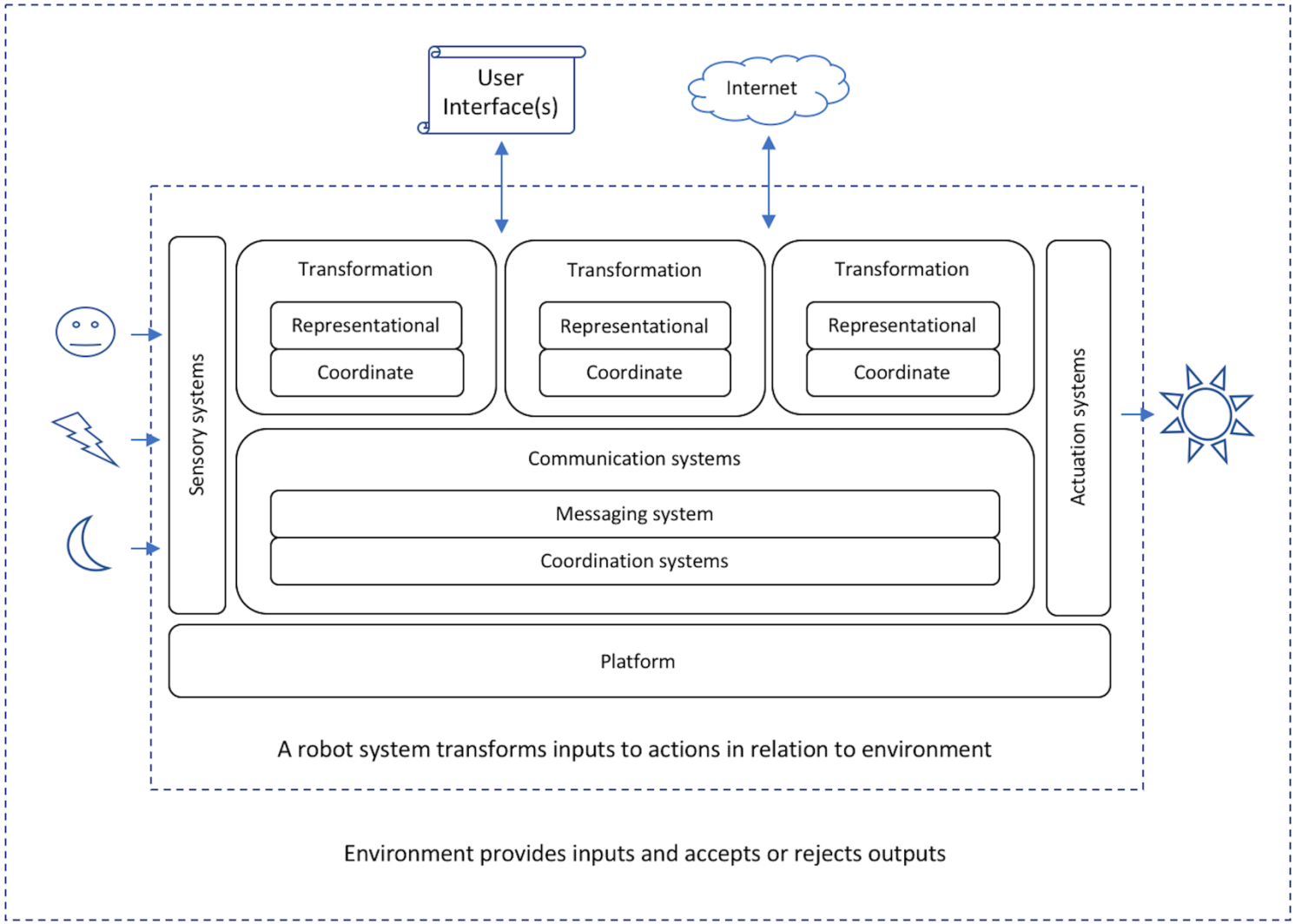

Figure 38: A contextually embedded model of a robot system (Own figure)

In terms of contextual embeddedness, the outermost band in Figure 38 represents the environmental context within which a robot system is embedded. As discussed above with reference to the visualisation and testing systems in Section 6.2.5, the surrounding environment provides inputs that condition the behaviour of a robot system, and that environment, in turn, either accepts or rejects the actions the behavioural model of a robot system produces. Thereby, the context in which a robot system is embedded represents the ground truth against which its performance has to be evaluated. Even if robot systems are expected to operate autonomously on their own, they do not do so as detached entities but with respect to given goals and the environment they are embedded in. This highlights the importance of the interactional and behavioural couplings between the robot system and its surrounding environment.

Interactional coupling refers to the capabilities of a robot system to interact with the surrounding environment. Robot systems differ from the ordinary screen-and-keyboard based computers in that their spectra of modalities for carrying out interactions is much broader, and they are more holistically and directly coupled with their surroundings. In addition to the physical embodiments presented above, robot systems are also equipped with user interfaces (B: CO7, CO27) and often connected to digital infrastructures and cloud-based services (B: CO6, CO12). Whereas sensors measure selected features and states of the surrounding environment, a user interface provides a human operator with means to interact with a robot system and monitor task execution. In addition, connectivity enables access to other services and sources of information, such as to mapping and object recognition data and services and other relevant information infrastructures. In turn, whereas actuators carry out physical actions by exerting forces, sensory and transactional data can also be fed back to different information systems and cloud-based services. Therefore, when conceptualising the interactional modalities of a robot system, it is necessary to include not only the physical embodiments but also other forms of digital and human interaction and information exchange. These different modalities of interaction in conjunction form a set of system boundaries in a broader and more distributed sense, taking into account different physical and digital domains with their respective purposes, affordances and constraints. The configuration of interactional modalities can be considered to define a degree of

interactional coupling between a robot system and its environment. In doing so, it also defines the information space within which the computational processes that produce the behaviour of a robot system operate. Therefore, while the modalities and degree of interactional couplings define the scope of interactions, they also define the informational boundaries and foundation upon which computer-controlled behavioural models can be constructed.

A behavioural coupling builds on the interactional coupling and is concerned with the capacity of a robot system to produce desired and appropriate behaviour. The notion of behavioural coupling refers to the extent to which a behavioural model is able to transform particular inputs to meaningful actions with respect to a given task and context. As discussed with reference to the communication and transformation systems in the earlier sections, behavioural models are produced by the interconnected computational processes that animate the embodiment of a robot system. As these computational processes reside in the middle of an embodied set of interactional couplings which are contextually embedded and bound, the degree of a behavioural coupling is not only context-dependent but also predicated on the degree of interactional couplings that mediate the interaction between the computer-controlled model of behaviour and surrounding environment.

The technological combinations that produce the interactional and behavioural couplings can be conceptualised as *chains of transformation*. This concept of the chains of transformation highlights the functional and organisational properties that characterise complex digital innovation in the context of robots and autonomous systems. The notion of transformations brings forward the presence of computational processes which transform inputs to outputs that are qualitatively different, whereas the notion of chains refers to the communication system whose purpose is to replicate data between different processes of transformation (nodes) without any distortions or delay. In conjunction, they present the structure and functioning of computer-controlled models of behaviour as distributed, interconnected and stepwise transformations whose purpose is to transform inputs into actions. This analytical difference between the processes of replication and transformation brings forwards and explains the differences reusability, composability and verifiability among different

software packages. Overall, the processes of replication appear more generalisable than the processes of transformation.

Furthermore, wrapping the layers of contextual embeddedness and embodiment around the chains of transformations shows that robots and autonomous system can be viewed as *contextually bound and embodied chains of transformation*. This addition seeks to highlight that the specificity of designs emerges from the contextual binding which renders the backdrop against which the interactional and behavioural couplings are constructed and evaluated. This gives rise to nested and multi-faceted arrangements that bear significant implications to the organising logic of complex digital innovation.

To conclude, Part 1 of this chapter has outlined an answer to the operative research questions that set out to identify the instances of subsystems and combinations and their respective characteristics. The process of thematic analysis condensed the documentary evidence to six salient themes. Of the six themes, five have now been described and discussed here in terms of their 13 constituent categories and elaborated to a model that conceptualises the structure and functioning of robots and autonomous systems. This structural-functional conceptualisation is one of the outcomes of this research. The subsequent Part 2 of this chapter shifts the focus on the primary research question and it is approached in the light of the conceptualisation presented in this part.

Part Two

The generative-integrative mode of development: From components to compositions

6.4 From components to compositions

This section examines how the tensions between the specificity of designs and distributedness of knowledge and control unfold in the development of robot systems. The purpose of this is to provide an answer to the principal research question, which asks how the tensions between the two can be resolved. The examination builds upon the themes and conceptualisation that have been developed in the preceding sections and combines them with the observations and data from the field notes and interviews.

The subsequent section describes and discusses the theme ROS community and software development and related categories. After this, the transferability and reusability of software packages are discussed before describing the systems development process that is conceptualised as a *generative-integrative mode of development*. The process sheds light on how the tensions between the specificity of designs and the distributedness of knowledge and control are reconciled. Then, the final section elaborates the role of under-specification and constructive ambiguity in this process before the closing of the chapter.

6.4.1 ROS community and software development

The theme of the *ROS community and software development* revolves around the software development in the context of robot systems, the reuse of software packages in the ROS community and the transfer of technological knowledge. One of the founding principles of ROS is to facilitate the reuse of robotics software in order to pool resources and share knowledge among the community members. As an open-source community, ROS facilitates voluntary participation in the absence of a central design agency and strict rules. Community members can make use of the ROS communication system and other software packages and resources as they see fit. Also, they can share their own code with other community members, maintain and fix bugs in existing software packages and contribute to the ROS ecosystem or domain-specific groups according to their interests (C: SE7, SE10, SE27).

This theme consists of two categories, the tools and infrastructure and the knowledge transfer. However, considering that the purpose of tools and infrastructure is also to transfer knowledge, these two categories are related. The analytical difference between the two is made on the grounds that the category of tools and infrastructure deals with the technological elements related of the software development and community, whereas the category of knowledge transfer is related more to the social and organisational elements.

The category of *tools and infrastructure* refers to the functionality and services that facilitate software development and distribution in the ROS community. As mentioned earlier, the listing[78] in the ROS wiki shows that approximately 2000 software packages are available through the ROS software distribution infrastructure, consisting of the ROS communication system, various transformation systems, hardware drivers, development tools and visualisation and testing systems.

The source code of different software packages distributed through the ROS infrastructure is not hosted in a single repository. While the source code of the core ROS and the central components and tools are managed and hosted by OSRF, the packages and source code that have been developed and made available by the community members are hosted in a distributed manner in the code repositories that may belong to commercial organisations, research institutes, universities and individual contributors alike. With this arrangement, the original developers can retain the control over their packages and source code even if they are willing to share and distribute it through the ROS infrastructure.

In order to make ROS distributions and packages available to the wider community, OSRF manages and runs a centralised build system[79] and software distribution infrastructure[80] (C: SE28). The ROS software distribution infrastructure gathers, compiles and distributes software from approximately 150 different repositories at the time of writing. To this end, the ROS framework

[78] www.ros.org/browse/list.php – the website that lists publicly available packages, which are distributed through the ROS infrastructure.
[79] wiki.ros.org/catkin – Build system and infrastructure for ROS.
[80] wiki.ros.org/build.ros.org – The public build farm is used to build binaries of the core ROS packages and any open source packages released by the community.

includes a custom-made build system that facilitates the compilation, packaging and distribution of code that resides in different code repositories and is written in different programming languages (C: SE13).

While the distributed hosting of source code shows how the knowledge and control of different technologies and software are distributed across the ROS community, it also shows that different software packages do not adhere to any single versioning system, particular release cycles or a project plan (C: SE16, SE29). ROS software is created and developed in a decentralised manner, yet, regardless of this distributedness of knowledge and control, the centralised build system brings together the code from different repositories making it available to the wider community in a centralised manner.

This has implications on the *characteristics of combination*. The decentralised development and asynchronous release cycles may affect negatively to the stability of software releases (C: SE29). OSRF develops and maintains the core components and functionality of ROS, including the communication and coordinate transformation frameworks and some development tools, and it avoids doing changes to the core components within the bounds of yearly releases[81] to ensure their stability and interoperability. However, the packages which are not the part of the core distribution tend to be subject to more or less frequent or unannounced changes. Their developers and maintainers (C: SE10) control the source code and carry out changes to their code according to their needs and plans (E: TR13). As changes to the code are picked up, compiled and shared by the centralised build and distribution infrastructure in a piecemeal manner, the software distributed through the ROS infrastructure can be said to follow a rolling strategy of release, which produces small and sporadic changes (C: SE29). Although this allows parallel and uncoordinated development, the lack of coordination may lead to instability and incompatibility if and when the packages which depend on each other are changed asynchronously. This rolling release of software packages that unfolds according to the needs and schedules of individual developers is not ideal from the point of view of robot system developers (C: SE29). In the end, the elements that constitute a particular

instance of chains of transformation should be compatible with each other for a robot system to function, and any unplanned and unannounced changes to any of the elements in the chain may prove detrimental.

To protect against inappropriate changes that remain outside the control of the developers of a robot system, the developers typically run and control their own robot-specific source code repositories, build systems and software distribution channels (C: SE13, SE16, SE22, SE23, SE28, SE29) for managing and compiling the code that is being developed and tested for a particular robot system. Separating a robot-specific codebase from that of the general ROS distribution allows the developers to evolve particular parts and stacks of robot-specific code gradually and according to their particular needs without it having any direct implications to the main distribution, unless the changes are incorporated into the source code the main branch. Furthermore, the organisations that develop productive applications on the commercial grounds may not want to share their core assets and intellectual property freely as open-source with their customers and competition (C: SE16, SE23).

Therefore, whereas robot system developers can make use and build upon the software packages distributed in the ROS community, they also must protect their codebase against untoward outside influences and guard their intellectual property to maintain the competitive advantage. This way, the public and private spheres of software coexist and evolve asynchronously. The publicly available software packages and source code are combined and integrated into compositions that are specific to a particular robot system, and every now and then selected changes can be pushed and incorporated to the main branch of the source code of a software package.

The category of *knowledge transfer* refers to the social and organisational efforts and initiatives to transfer knowledge concerning software packages, their underlying technologies and potential scenarios of use. To consolidate knowledge on software packages and robot software development, the websites wiki.ros.org, answers.ros.org and the ROS-related discussion forums provide

[81] wiki.ros.org/Distributions – A ROS distribution is a versioned set of ROS packages. The purpose of the ROS distribution is to allow developers work with a relatively stable codebase. Therefore, once a distribution is released, OSRF tries t o limit changes to bug fixes and non-

the venues for documentation, peer support and general discussion. Wiki.ros.org provides a place for the general documentation of ROS, software packages, tools and methods in terms of their functionalities and installation, and answers.ros.org is a discussion forum that provides peer-support concerning the technical implementation challenges. In addition, some of the common conventions used within the ROS framework are documented in the form of ROS Enhancement Proposals[82]. In turn, general discussion regarding the ROS community takes place on discussion forums and email lists, whereas the ROSCon conferences bring people together on a yearly basis.

In addition to the community-wide venues of knowledge transfer, several local events and more targeted activities are also organised. Local events and training serve regional and more targeted needs, and there are also multiple subdomains and interest groups (C: SE26, SE27). They seek to pool their resources and knowledge around particular areas of interest, for example around specific application domains such as industrial manufacturing, agriculture and military, or around particular technologies such as quadrotors, hardware drivers, ROS2 or running ROS on embedded devices. This way, the category of knowledge transfers represents the social and organisational side of the ROS community as the community seeks to make ROS more readily usable and accessible to the wider audience.

6.4.2 Transferability of software and knowledge

The observed diversity of the ROS community raises a question to what extent different software packages are readily transferable across different robot system or domains of productive applications. In the end, as discussed earlier, roboticists work on a rainbow of use cases and environmental contexts. This renders the community highly heterogeneous while the knowledge and control of different technologies are distributed among the community members without any centralised design agency. This section sheds light on this matter by drawing on the observations and discussions in the events and interviews as listed in Appendices H and I.

---

breaking improvements for the core packages (everything under ros-desktop-full).
[82] www.ros.org/reps/rep-0000.html - Index of ROS Enhancement Proposals (REPs).

The extent to which software packages are transferable across different robot systems and domains of productive applications vary significantly. This matter was often discussed along the lines of the dependency on physical embodiments, the contextuality of behaviour and the partitioning of chains of transformation as well as the underlying objectives of robot software research and development.

The dependence on the physical embodiments was one of the commonly mentioned aspects in relation to transferability. The physical groundedness of data and computation is condensed in the quotation from an interview below.

"[G]athering experience means that you need to have an embodied system for your data to be somehow meaningful." (I: IN1)

The data a robot system processes and operates upon represents its embodiment, environment and behaviour in terms of a variety affordances, capabilities and constraints. Thereby the experimental and observational data a robot system produces for the purposes of research and development is highly embodied and context-dependent. This has implications to the transferability of research results, as expressed in the quotation below:

"I think the big challenge that robotics has at the moment is reproducibility. Because, a lot of the time people developing their own systems, their own hardware as well as their own software independently of each other, and it makes total sense to do that because you have the expertise and you are looking at the particular effect, but it makes it very hard to share things between different labs. So, to be able to create a larger and more integrated system that is able to solve more than the particular problem that you are trying to solve, that's where the challenge is, how to integrate, how to share results." (I: IN1)

However, chains of transformation are developed not only with reference to physical embodiments of a robot system but also with reference to tasks and task environments. They implement behaviour and functions that are derived from the characteristics of a task and from the environment in which the task is going to be performed, and the fitness of any particular behaviour is defined by the overall context in which the performance takes place.

In order to make software transferable, the question remains on how and according to which logic should contextually bound and embodied chains of transformation be partitioned to generalise different pieces of the behavioural logic so that they could be transferred and recombined in order to produce other robot systems? When the overall behavioural model developed with reference to a particular robot system is partitioned into separate algorithms and software packages, the resulting packages tend to come to carry the embedded underlying design criteria and implicit system-wide assumptions. This way, software packages that originate from different sources do not necessarily share the same underlying design assumptions or quality criteria (B: CO1, C: SE03, SE9, SE11, SE12). The assumptions may differ for example in terms of the affordances and limitations of physical embodiments, the semantic meaning and use of message types in the context of the ROS messaging system, expected service levels for real-time operations, error handling, the level of testing or documentation among others. All this may hinder the reuse of packages when they are collected and distributed in a piecemeal manner from heterogeneous sources. Against this backdrop, it is not a surprise that the challenges on the replication and reproducibility of research results and the reusability of code have recently become under scrutiny in the field of robotics (Antonelli 2015). Researchers have started calling for greater transparency on testing methods to ensure the reproducibility of research results (Bonsignorio & del Pobil 2015).

The objectives of robot software research and development may also hinder the production of transferable software. When the distributed software derives from an academic research project, the authors of the code may not exert excessive efforts to abstract and generalise the code to maximise its transferability and reusability (B: CO1, SE9, SE11, SE12). In the end, the objective of a research project is often to produce some novel technology, algorithm or a proof of concept which can be published in an academic journal. Researchers are rarely incentivised to deliver and maintain readily transferable, robust and reliable code. Furthermore, it is important to note that the development of algorithms and the engineering of software are different problems. When probed by the author in a robotics conference reasons for this, the answer was *"you cannot create software if you do not have an algorithm first"*. This was also echoed in one of the conference presentations (B: SE9) which highlighted that while

graduate students may generate a lot of code, they are not software engineers and often do not maintain the code after graduation. However, others may pick up the previous work if they find it useful and develop it further according to their needs. Regardless of this, it must also be noted that some research laboratories and institutes seek to develop and package their software in a way that makes their contributions more readily transferable and reusable, this being especially the case with domain and function specific software packages (E: TR2, TR5, TR13) where the overall structuring of tasks and task environment is somewhat known (C: SE26, SE27).

In this light, it is not surprising that robotics researchers are often more focused on the development of algorithms than software engineering, and that the scope and quality of the distributed software packages may vary to a great extent. Whereas research robots are created to produce novel technologies and proofs of concepts, turning them into productive applications and commercial innovations often requires further development to increase reliability, robustness and transferability (SE8).

Regardless of the challenges surrounding the transferability of research outcomes and software packages, ROS clearly demonstrates that to a certain extent software can be transferred across robot systems and use cases. To explore this further, it is beneficial to take a look into the stability and centrality of software packages in terms of the themes and categories presented in Part 1.

According to the Community Metrics Report[83], ROS software packages related to the communication system and development tools are most downloaded. This is not surprising since the ROS communication system provides the common methods and tools for constructing a robot system as a distributed computer, and this is where the many of the needs and requirements from different domains of robotics converge. Thereby, the packages that constitute the communication system show a great degree of transferability and reusability. This seems to apply also to the software development tools and methods that are used in the visualisation and testing of robot systems.

[83] wiki.ros.org/Metrics – ROS Community Metrics Reports.

Whereas the communication systems are used to establish interconnections between computational processes, the coordinate transformation systems are used to define and compute the structure and articulation of the physical embodiment. The needs and requirements in this area also converge to an extent, and these central packages are also developed and maintained by OSRF. The robot model (urdf) and the library for performing coordinate transformations (tf) provide the methods and tools to define the structural elements, relationships and properties of the physical embodiment that compose a robot system and carry out coordinate transformations. While these packages are transferable and reusable (E: TR1, TR3, TR8) to a large extent, they are not applicable to all scenarios. Therefore, alternative models and methods have been developed and made compatible with ROS (E: TR9, TR14, TR15) to overcome the limitation of the core packages. This way, whereas the coordinate transformation systems are also transferable and reusable, they show a greater degree of variety than the communication systems and development tools.

In contrast to the messaging and coordination transformation systems, the physical embodiments and systems of representational transformations show much greater variety. As discussed in Part 1 with reference to the physical embodiments, hardware makers produce a variety of sensors, actuators and hardware platforms that offer different modalities of interaction in response to the variety of tasks and task environments. Similarly, there is also a plethora of algorithms and software packages available to produce a variety of representational transformations that might be needed when developing a particular model of behaviour for a robot system.

Based on these observations, the needs and requirements tend to converge in relation to the communication systems, development tools and coordinate transformation systems while they diverge on the physical embodiments and representational transformation systems. This indicates that the reusability and transferability of software packages are not simply a binary matter, but vary according to the purpose, functional characteristics and origins of a package.

6.4.3 Generative-integrative mode of development

Based on the conceptualisations, themes, categories and characteristics of combination presented in the previous sections, this section presents and outlines the concept of *generative-integrative mode development* to characterise the process of robot system development.

Although ROS facilitates the interconnection, distribution and reusability of software packages, it does not enforce any particular methods or standards on how distributed computational processes should be partitioned, arranged or interact with each other (B: CO1). While there are some common conventions, ROS provides the developers with flexibility to proceed in a way that best suits their particular needs. To an extent, packages can be combined in different ways and the open source code allows developers to modify the code when that what is already available does not serve their needs. While this flexibility is often brought forward as one of the main benefits of ROS (H: EO4), the lack of standards is also seen to hinder the efforts of reuse (H: EO3, EO9, EO10).

Moreover, while ROS and other robot software development frameworks are frequently referred to as and discussed in terms of platforms, they typically lack the essential characteristics of a platform. In particular, they are not platforms when a platform is understood as a singular stable core component that provides clear boundary resources, interfaces and rules upon which applications could be built and components connected (Eaton et al. 2015; Baldwin & Woodard 2008). ROS is not an operating system in the traditional sense of the word but a framework for developing robots and autonomous systems as distributed computation.

In this context, software development can be seen to unfold as a process that can be labelled as a generative-integrative mode of development. As discussed in Section 3 (6.3), the design and development of a robot system begins from the evaluation of the context. To equip a robot with sufficient interactional modalities, an appropriate physical embodiment has to be selected or constructed. After that, the development of a behavioural model as chains of transformation can proceed. To construct these chains of transformation, the developers often seek to reuse and combine existing software since it provides

access to a wide body of technological knowledge while decreasing the development time and the need for development resources.

The development of meaningful and reliable behavioural models requires careful combination and integration of the physical embodiments, communication systems and transformation systems. A common approach to begin the development work is to combine the first version of a robot system from the software packages that are already available and readily interoperable (RS18). In this approach, the developers first carry out the initial design and functional (de-)composition based on the task and task environment specific requirements, and then identify the functions and packages which in combination are seen to provide an appropriate starting point for further development. Subsequently, the developers combine these different packages to create the first working version of a robot system. While this generative combination produces the first version, it cannot be considered as a finished product. Instead, it serves as a starting point providing the embodied system for experimentation and data collection that is necessary for further development.

With the first version in place, the developers can start working on the integration of the behavioural model and focus primarily on the areas that are seen most beneficial in terms of the objectives of their project. These integration efforts can be viewed in terms of internal composition and outward behaviour since the developers attempts to integrate the behaviour of a robot system better with its task and task environment are reflected in the internal composition and interactions within the chains of transformation. Although the integration occurring after the generative combination seeks to produce a well-functioning and integrated whole, it is not always a straightforward process and may require significant modifications to the initial combination. Figure 39 illustrates figuratively how Team Delft combined and integrated the components from the MoveIt! motion planning and controlling package with their custom-made components to come up with the chains of transformation that served their particular purposes in the Amazon Picking Challenge.

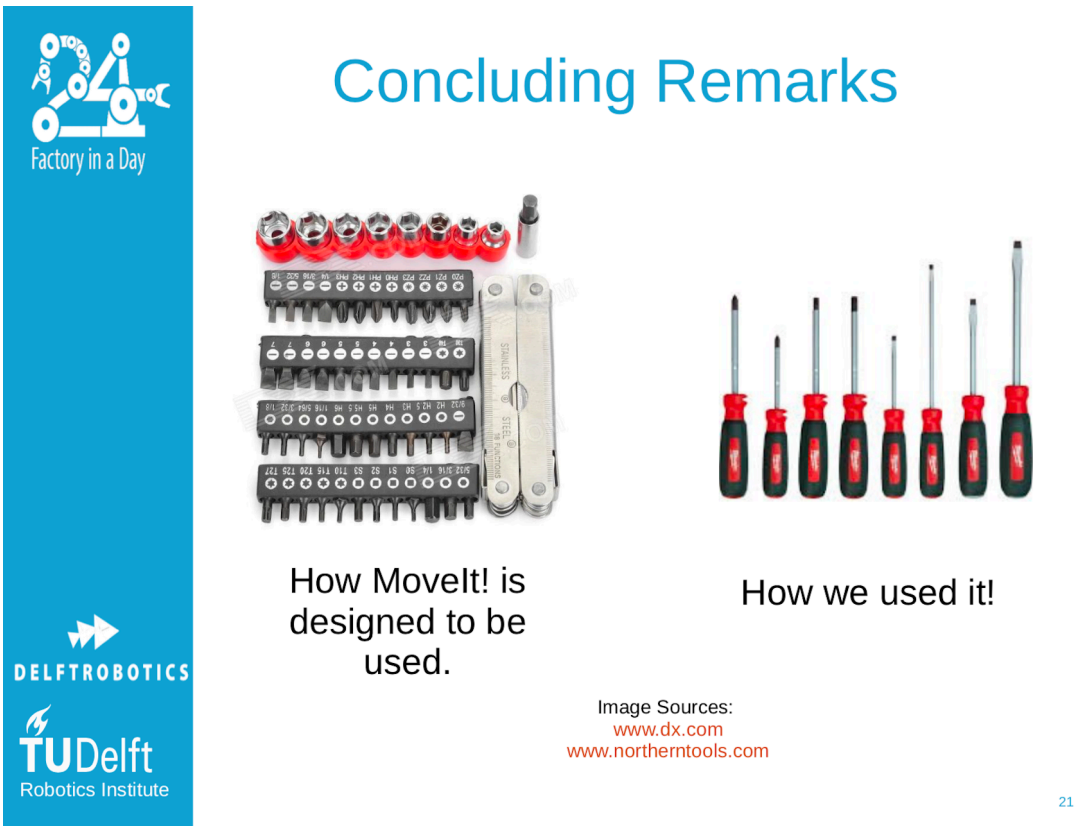

Figure 39: Team Delft's implementation of MoveIt! for Amazon Picking Challenge (D: RS19)[84]

Moreover, while ROS provides a number of reusable and transferable packages, the cost of search and verification increases as packages become more purpose-specific (H: EO4). Finding and testing software packages requires time and resources even if the packages would be nominally free of charge. They may not always be well documented and might be of low quality, requiring further debugging, development and testing or even complete reimplementation of the underlying idea (EO3, EO4). Furthermore, as different software components and packages do not necessarily share the same underlying system-wide design assumptions their integration might be challenging or even impossible in some occasions, even if they appear to be compatible at a first glance. Of course, much of this can be overcome by rewriting a package or modifying selected parts of the code if the source code is publicly available, but this should not be overlooked as it may require significant time and effort.

[84] Republished from the 2016 ROSCon presentation Plan to Win with MoveIt! - Lessons learnt from the Amazon Picking Challenge 2016, copyright (2016) Mukunda Bharatheesha, with permission. Images on the slide are from dx.com and northerntools.com.

The role of underlying assumptions was also brought forward by the Team Delft as they summarised the lessons they learned when preparing for the Amazon Picking Challenge. They condensed the challenges of the contextual binding and implied assumption as follows: *"Gravity is a heartless entity!",* and *"Assumptions bear the roots of all disasters!" (D: RS19).*

In a similar manner, the integration challenges related to the generative combination were highlighted by a robotics researcher in one of the interviews:

"If you build an integrated system, then [you] tend to know what all the different kinds of components are doing. Some people think that you can plug things just together and get some emergent behaviour come out. Maybe it's true and maybe it's not. It depends very much if you are lucky or not." (I: IN1)

However, another interviewed researcher (I: IN2) told that she used ROS-based software and components as black-boxes concentrating only on the behaviour of the algorithms that were of her primary research interest. Although the rest of the robot system offered her with an embodiment against which to carry out her research, she emphasised of not being in the business of developing robots but studying the behaviour of particular machine learning algorithms.

The reusability and compatibility of different software packages emerged frequently throughout the research. In the end, the both version of ROS have been designed to facilitate the reuse of robot software, yet it does not enforce any particular standards (B: CO1). As long as the computational processes agree on the message type, paradigm and name of the connection, they are able exchange messages, yet the ability of any two nodes to exchange messages does not imply that they share the semantic meaning of the message or that a syntactically valid and successful interaction would produce meaningful behaviour at the system level. The difference between the syntactic and semantic compatibility and the system level functionality is manifested by the prominence of the visualisation and testing systems. Even if a robot system would be functional and working in a purely syntactic and technical sense, its behaviour with respect to its task and task environment can be lacking. In addition to the syntactic compatibility, the semantic compatibility among the interconnected transformations needs to be increased so that the transformations in combination would produce the desired behaviour.

Robot system developers evaluate the level of behavioural coupling and semantic compatibility using simulations and physical robots. With the simulation systems, developers are able to run controlled experiments in a cost-efficient manner before beginning the testing with the physical robots. As the simulated worlds are compositions of computational simulation objects and models, their usefulness and reliability are predicated on how closely they have been crafted to resemble and model a particular task and task environment. Regardless of the inevitable reality gap, they provide the methods and tools to evaluate and examine different models of behavioural and to work towards the higher levels of semantic compatibility. Once the appropriate levels of behavioural coupling and semantic compatibility have been reached in simulation, testing and development can proceed with the physical robots in the actual task environment. As the actual physical environments are more open-ended than their simulated counterparts, it is not uncommon for a robot system to perform better in simulation than in the actual environment. However, while the open-endedness of the physical environment provides the environmental and operational variety necessary for well-grounded testing, it is also challenging to reproduce (F: VI15). To ensure the reliability and robustness of behavioural models over a range of situations, testing and development in both simulated and physical environments are often needed.

Based on these observations, the generative-integrative mode of development can be conceptualised as a process that begins with the initial generative combination of the physical embodiments and software packages and then continues with the subsequent iterative and cyclical process of gradual integration with an aim to compose a robot system that produces meaningful behaviour. Once the first working version of a robot system is functioning, it can be used to gather experiences and observations that are necessary for composing a well-crafted and dependable model of behaviour.

6.5 Under specification and constructive ambiguity

The generative-integrative mode of development illustrates the challenges that revolve around transferability and reusability of software packages, yet the wide uptake of ROS demonstrates that this does not prevent from making use of what is already available. As discussed above, ROS does not enforce any particular

standards. The software packages that originate from the community may not be fully compatible with each other, even if they would be able to exchange messages with each other through the ROS communication system. Therefore, the question remains on what grounds software is then transferred and reused? The tentative answer to this question is the under-specification of interconnections which can be viewed as constructive ambiguity, even if this may seem as a questionable practice.

The interconnections and operational run-time messaging between the ROS nodes are performed through the messaging system. For this to happen, the coordination system starts nodes, making them to establish interconnections according to a system configuration that is specified in a start-up file. The composite behaviour that is produced through the distributed computation materialise only at the run-time when the system is operational and the interactions between the nodes occur. In turn, when the system is stopped, and no interactions between the nodes occur, its systemness disintegrates to a set of unconnected software components until the interconnections are reanimated.

The interconnections between the nodes are based on data exchange according to a particular message type and method of communication. The specification of these inter-node connections can be viewed as under-specified and partial considering that the syntactic specification of a message type and method of communication does not extend to the semantic, contextual and behavioural aspects of a composition at the level of a robot system. Even if there are some common conventions[85] and practices that inform how robot systems can be designed and developed to ensure broader compatibility and interoperability among software packages, in general, the lack of the explicit specification of various non-interface design parameters does not expose the system-wide and contextual assumptions that may underlie the design of a particular component. As the interconnection-level specification often does not capture all design variables and assumptions, the compatibility over the interconnection can be viewed as under-specified.

[85] www.ros.org/reps/rep-0000.html – Index of ROS Enhancement Proposals (REPs).

The under-specification enables constructive ambiguity as it allows developers to overlook certain non-interface design parameters. While this facilitates the production of generative combinations, it does not necessarily produce well-behaving systems. The under-specification often necessitates additional design, development and testing in order to craft and integrate components into a composition that satisfies specific requirements of behaviour and robustness, yet these multi-faceted engineering and adjustment efforts tend to be convoluted and may still not result in the desired outcome (H: EO11). Whereas this flexibility can be leveraged and managed in the context of research and development, it poses challenges in the view of mission-critical and commercial applications. The lack of precise specifications rarely renders the level of reliability and verifiability that is required in mission-critical and other high-performance applications.

The benefits and drawbacks of under-specification emerged in various occasions and guises during the course of the research. It was entirely possible to hear completely contradictory commentaries concerning the profitability of the under-specified approach. While others highlighted that the reuse and transferability had reduced significantly the costs of software development (I: IN4), others pointed out that simply circulating and amending pieces of software around does not create a solid foundation for software reuse nor does it encourage the use of systematic software development practices as developers tend to gravitate to start from what is already available regardless of its quality and usefulness (H: EO4). This was summarised by a robotics professor (H: EO8) along the following lines:

"ROS is standard, and everybody uses it. It's a pragmatic approach for research as everybody knows it and it provides the basic architecture. However, if you ask 4 people about ROS, you get 4 opinions. But there is not really anything else widely available at the moment."

While the developers of robot software frequently refer to the practices of general software engineering and highlight the benefits of information hiding, abstraction, platforms, interfaces, application stores and software reuse, at the same time, they are cognizant of the distinct requirements that characterise the contextually bound and embodied chains of transformation. Even if the need for reuse and commercialisation of applications and software components are well-

recognised, the application stores were not seen as an appropriate analogy for several reasons (H: EO4). For example, whereas ordinary desktop and mobile applications operate individually on the top of a platform (i.e. an operating system), the various components of robot control software interact with each other in a distributed manner, thereby complicating the incorporation of any new or altered components into existing compositions. Other obstacles were brought up as well, such as the lack of standard interfaces, the lack of detailed knowledge regarding what some particular component actually does, the absence of appropriate test cases and occasionally poor documentation (H: EO4, EO5). Against this backdrop, some rejected the idea of software as black-boxes. They preferred grey-boxes with the access to open source code and the internal representations and intermediate states of data to be able to evaluate and examine the functionality and performance of software components at a necessary level of detail.

Moreover, discussions frequently returned to the core problem of how to identify and define suitable levels of abstractions; finding the "right" primitives and levels of abstraction were often seen as the main obstacle (H: EO11). Regardless of multiple efforts, the semantic modelling of various aspects of behaviour and environmental contexts remain challenging. In principle, for modelling efforts to succeed, a priori agreement should be reached on how to partition and define elementary motions, movements and behaviours across different levels of abstractions, ranging from low-level control software to higher levels of perception and control as well as to the descriptors of characteristics and behaviour of the surrounding environment. The partitioning of chains of transformation and the detaching of the physical embodiments from the surrounding environment were seen problematic without well-defined abstractions and meta-models which would define pertinent system-wide aspects of contextual and embodied behaviour and interaction.

The importance of well-rounded interface specifications was brought up in a robotics summer school (H: EO6) where the following statement was made: *"abstraction is key but [information] hiding is dangerous"*. While software standardisation remains challenging, there are initiatives whose objective is to develop meta-models that describe skills and behaviours as compositions of

tasks, object affordances, robot capabilities, environmental contexts and other relevant constraints that need to be solved (H: EO6). In a way, computer-controlled behaviour can be described as a constrained optimisation problem, and this brings forward the importance of defining the constraints and boundaries at different levels of abstraction within which the computation of behaviour takes place. However, categorising and subdividing the world and its motions and behaviours in their various intricacies into computable pieces is far from unambiguous, especially if the domain to be modelled is not well-defined in advance. Some are sceptical concerning the attempts of modelling and standardisation, and they would prefer a more pragmatic approach where beneficial conventions and standards would emerge through the practice.

To overcome some of these challenges, different domain and technology-specific sub-communities seek to pool their resources and focus their efforts on more narrow problems and domains, such as industrial, aerial or agricultural robotics, to make progress towards their particular goals and needs (C: SE27) as this allows for the implicit specification and agreement of underlying assumptions. Although the broader robotics community remains divided on to what extent generic methods of modelling are feasible, ROS with its under-specified interconnections has nevertheless gained traction, portraying that the process of the development of a robot system can unfold as a generative-integrative manner.

In the light of the empirical findings and conceptualisations presented in this chapter, it appears that there is no straightforward answer or process according to which the tensions between the specificity of designs and the distributedness of knowledge and control across the community can be resolved. The observed approach depicts itself as a multi-layered generative-integrative process. The developers a robot system construct the first version of the system from a diverse set of software packages and components, after which this initial combination is further developed and integrated into a desired composition in a way that is most feasible in given circumstances. Overall, the production of a dependable and robust composition requires careful integration, yet all-encompassing a priori attempts of exhaustive specification appear to remain elusive. The domain-oriented design and development approaches seem to

overcome this problem partially as they share similar underlying assumptions regarding the contextual and behavioural requirements.

6.6 Summary

This chapter consists of two parts and describes how the tensions among the specificity of designs and the distributedness of knowledge and control are reconciled in the context of ROS.

Part 1 describes and discusses the themes and categories that emerged from the thematic analysis and develops a model that conceptualises robots and autonomous systems as *contextually bound and embodied chains of transformation*. This conceptualisation brings forward the functional and structural characteristics which underpin the organising logic of complex digital innovation in the view of product architectures and characteristics of combination.

Part 2 takes a closer look into the organising logic and explores how the tensions between the specificity and distributedness unfold and are resolved during the development of robots and autonomous systems. Against this background, the observed process of development was conceptualised as the *generative-integrative mode of development*. It describes a process where the first version of a robot system is generatively combined by bringing together different physical and digital components. This generative combination is then followed by the phase of integration. During the integration phase, the behaviour of a robot system is experimented, observed and adjusted, and the combination of components is gradually crafted into a composition that produces the desired behaviour with respect to a given task and context.

7 Discussion

The purpose of this chapter is to discuss and establish the conceptual findings presented in the previous chapter. Based on the empirical findings, the previous chapter developed two novel conceptualisations to characterise the organising logic of complex digital innovation. In order to establish their validity, boundaries and applicability, they are evaluated in the light of the extant literature that conceptualises the organising logic of innovation in the view of product architectures and combination.

The first section of the chapter restates the conceptual findings to summarise their salient characteristics. Subsequently, the second section enfolds them in the literature by discussing the proposed concepts with reference to the related literature that was reviewed in Chapter 2. After the novelty of the proposed concepts has been established, the third section elaborates how they contribute to the literature on digital innovation and discusses briefly their wider applicability and relation to the literature on simulations. Potential avenues for further research are discussed before the concluding remarks.

7.1 Summary of conceptual findings

Chapter 6 brought about two novel conceptualisations. The first one of them presents a structural-functional model that conceptualises robots and autonomous systems as contextually bound and embodied chains of transformation. The second one characterises the process of system development as a generative-integrative process where generative combinations of components are crafted into compositions. These conceptualisations are restated below before proceeding to the enfolding literature.

7.1.1 Contextually bound and embodied chains of transformation

The structural-functional model conceptualises robots and autonomous systems as *contextually bound and embodied chains of transformation*. This captures the underlying logic of complex digital innovation as it intertwines the physical, digital and behavioural aspects that characterise complex digital systems and

innovation. This conceptualisation seeks to differentiate complex digital innovation from other domains of digital innovation which cater to different purposes and adhere to different operational principles and organising logic. The purpose of this is to provide a conceptual lens and clarity for discriminating between the empirical findings that emerge from different empirical contexts.

A good starting point to begin unpacking this conceptualisation is the overall purpose and functioning of robots and autonomous systems. In short, they are expected to produce goal-directed and context-dependent behaviour in order to be able to operate autonomously with limited human intervention. Once set in motion, robot systems transform sensory inputs into actions according to their physical structures and computer-controlled models of behaviour. Therefore, it follows that a defining characteristic of robots and autonomous systems is that their behaviour emerges from the direct interaction with the surrounding environment. This way, a robot system is operated by the environment in which it is embedded as the environmental inputs condition its behaviour, whereas, at the same time, the human operator becomes moved further away from the control of situated action. The environment that renders inputs also provides the ground-truth that either accepts or rejects behavioural outputs, rendering the criteria against which the fitness of the interactional and behavioural couplings will be evaluated (Alexander 1964). To reach a sufficient degree of fitness, the developers of a robot system are expected to join the physical embodiments and communication and transformation systems in a way that produces appropriate interactional and behavioural couplings that are congruent with tasks and tasks environments.

In functional terms, the notion of contextually bound and embodied chains of transformations bears significant importance, as it highlights that the inputs and outputs of a robot system differ in qualitative terms. To produce purposeful real-time behaviour, a robot system measures constantly certain features of the surrounding environment and transforms them into actions and actuations which exert forces onto environment to cause a change in some state of affairs. This is radically different when compared to the information and communication systems, which are predominantly used to mediate human communication and replicate digital objects across time and place – such

systems could be viewed as chains of replication. However, the distinction between the replication and transformation as the opposite ends of a spectrum should be seen as analytical. In practice, many systems are located somewhere between the two ends of the spectrum. For example, an accounting information system may transform individual financial transactions into consolidated summaries that report the overall loss and profit (Klaus et al. 2000). However, while they do so, they do not act based on the reports they produce, keeping the interpretation and subsequent actions based on the reported results in human hands. Leaving a robot system to its own devices shifts the focus from replication to transformation, from the mediation of information and communication to situated action and behaviour.

This bears significant implications on the design, reliability and verifiability of systems, among other things. The detailed specification of a system's behaviour remains challenging considering the variety of environmental and behavioural variables that would need to be formalised and modelled to make them computationally tractable. Occasionally, controlled environments are constructed around robot systems to make them more manageable by containing the variety and contingency that abound in open-ended environments. While this is the case with the production lines in factories, there are numerous application domains where this is not feasible. For example, the infrastructure and system of road transit with all its roads, cars and drivers among other things cannot be encased for self-driving cars to be able to operate. In these cases, robot systems must be developed with an aim to ensure that their physical capabilities and behavioural models can handle the variety of environmental and operational conditions. This has also implications for the verification of robot systems. In the end, whereas the correctness of a replication system can be established by evaluating the degree of similarity between the inputs and outputs of a system, it is much harder to establish and evaluate if large sets of transformations produce acceptable behavioural outcomes with respect to a goal and context in a range of situations. Establishing the correctness of the relationships among a range of input and output transformations is much less straightforward given the variety and intricacy of the relationships among environmental conditions and meaningful transformations (Bonsignorio & del Pobil 2015). Often, an excessive amount of

testing is required in simulated and physical environments to verify the behaviour of the chains of transformation under different conditions, especially when a task environment is open-ended and unstructured.

In structural terms, this conceptualisation seeks to describe what constitutes the embodied chains of transformation and how this affects to the organising logic of innovation in the view of combination and product architectures. As shown in the previous chapter, such artefacts are made of physical embodiments and the systems of communication and transformation which in combination render the interactional and behavioural couplings that produce the overall behaviour. This brings forward the distributed character of computation highlighting that the overall system-level model of behaviour is composed of individual transformations that interact with each other. These embodied chains of transformation can be (de-)composed in multiple ways and they grow in complexity as the number of interconnected components, sensors, actuators and processes of transformation, increases, and they are can be viewed and implemented as sets of distributed computational processes. These sets can take multiple forms and the ROS communication system facilitates the setting up of such sets in a flexible manner. Therefore, one of the defining characteristics of ROS is that there is no one particular central platform (Baldwin & Woodard 2008). Instead, a robot system usually consists of a variety of bespoke and distributed computational arrangements, which carry out the stepwise transformations in different ways depending on the goal, context and composition of a robot system. This shifts the focus of conceptualisation of robot systems from central platforms and stable interfaces to distributed computation and multidimensional connectivity.

This complex and distributed character of computation bears implications on the practices of digital innovation. Stable platforms, interfaces and application stores are often referred to as an ideal target state regarding the reusability and distribution of software. However, while software packages are shared and reused among roboticists, the application stores remain absent. Some question the appropriateness of this analogy in the context of robots and autonomous systems given the fundamental architectural differences between the typical platform-like computer operating systems and the systems that produce their

behaviour through distributed computation. With traditional operating systems, applications run on the top of the operating system as relatively independent entities. In turn, in the case of distributed computation, various processes and components that may run on different computers produce the overall behaviour in combination. This means that changing the behaviour of a single component often influences the overall behaviour of the whole system. The distributedness challenges the notion of stable platforms and interfaces.

The notion of the chains of transformation also brings forward the implications in the view of the transferability and reusability of software. While the ROS communication system demonstrates that the systems that interconnect different processes of transformations can be generalised in a way that they cater the needs of the broader robotics community, the transferability and reusability of particular transformations is more limited, being contingent on system architectures, tasks and contexts which set boundaries to their generalisability.

The notion of contextually bound and embodied chains of transformation captures the salient characteristics of complex digital innovation by showing how the intertwined characteristics of the physical, digital and autonomous aspects that have implication to the organising logic of innovation in the view of product architectures and combination.

7.1.2 Generative-integrative mode of development

Roboticists seek to build upon each other's work by reusing existing components, yet the empirical observations demonstrate that this is not always a straightforward process. The analysis of the empirical observations produced a model that conceptualises the process of robot system development as the *generative-integrative mode of development* that involves under-specification and constructive ambiguity. This model offers an answer to that principal research question that asks how the tensions between the specificity of designs and the distributedness of knowledge and control can be resolved. The generative-integrative mode of development can be described as an approach in which an initial generative combination of components comes to provide a starting point for further systems integration efforts. This way, the integration

follows the combination, and during the integration phase the behaviour of particular chains of transformation is experimented, observed and adjusted. The combination of components is thereby gradually crafted into an integrated composition that is expected to produce the desired behaviour with respect to a task and context. This process is elaborated below.

A developer begins the work by carrying out a functional decomposition, looking into the physical embodiments and sets of transformations that would produce the desired model of behavioural. A set of components is then selected and combined using the resources that are readily available, such as hardware components, software packages and frameworks. If suitable components and functionalities are not available, the developer must create them. Subsequently, components are combined into a robot system. This initial combination can be characterised as generative as it draws from the end-product agnostic components that originate from distributed and heterogeneous sources. While this combination produces the first working version, this first version rarely fulfils the requirements. Instead, it provides a starting point for further integration efforts. With a functioning system in place, the developers can experiment, observe and gather data that is essential in the further improvement of performance, reliability and robustness of the system.

Whereas generative combination speeds up the early phases of development, the subsequent integration phase requires further efforts. The degree of efforts depends on the extent to which the resulting combination of components fulfils the initial requirements, for example in terms of behaviour, reliability and robustness. Depending on the requirements and shortcomings, additional development can take place along various lines, such as by altering the configuration of hardware, improving the system level integration and error handling or by developing new transformation systems altogether that facilitate the production of specific behaviours.

The complexity and amount of details and variables associated with the contextually bound and embodied chains of transformation limits the feasibility of exhaustive a priori specification. Often, much experimentation, adjusting and testing are needed to establish not only the appropriate behaviour of a system,

but also to identify and establish the reliability of the environmental features upon which the functioning of a behavioural model can be conditioned. To this end, the post-combination integration efforts make extensive use of virtual simulations environments and physical experiments. They provide data on the behaviour of a system in various environmental conditions to examine and evaluate robustness and dependability. Often, extensive integration and testing are needed to bring behavioural models to a level that produces required behavioural coupling and congruence. This way, with the generative-integrative mode of development, the work begins from a generative combination, which is then crafted into a composition. This shows that simply bringing existing components together is not likely to produce the desired behaviour.

Moreover, components which originate from different sources and serve different functions vary in terms of their transferability, reusability and quality. Whereas the components that cater to common needs and use cases are more readily transferable, such as the ROS communication and coordinate transformation systems, others might be significantly less so. This is, for example, a case with the components that produce specific representational transformations for narrow use cases. Furthermore, considering that components often emerge from different origins in the absence of centralised design agency, they do not necessarily adhere to the same underlying assumptions and design principles. While this does not automatically prevent from bringing them together, the absence of the overall systems level design principles may cause problems when the components taking part in distributed computation react to some system events in an unspecified or uncoordinated manner. The existence of a syntactic interconnection does not imply the existence of a semantic integration in the view of the overall system architecture. These things are such as what should be a default behaviour in case of a sensor failure or in the malfunctioning of some of the computational processes, among many others. Leaving the underlying assumptions, such as the affordances or constraints of the surrounding environment and physical embodiments, underspecified may lead to decreased levels of dependability. In this light, the under-specification that allows for the generative combination can be seen as constructive ambiguity.

The high degree of systemic interdependencies combined with the lack of centralised coordination and uncoordinated design practices leads to a situation where components and frameworks cannot be viewed as black-boxes from the point of view of systems integration. Considering that all parts of a distributed system are expected to work together in an integral manner, the developer of a robot system is left with the task of establishing the overall system level design goals and principles, which were not present when the end-product agnostic components were designed and created (Yoo et al. 2010; Yoo 2012b). In this view, the process of integration refers to the efforts and actions that are taken in order to raise the level of integration among the components of a robot system (Tolk et al. 2007) by removing the gaps and incongruences caused by the under-specification.

The generative-integrative mode of development expands the focus of digital innovation from generative combinations (Yoo et al. 2010; Yoo 2012b) by incorporating the iterative process of integration during which the initial combination is elaborated into a composition. Different components are evolved in conjunction to develop contextually bound and embodied chains of transformation that produce desired and dependable behavioural models. This process requires architectural knowledge which is absent when components are developed without a centralised design agency (Nambisan et al. 2017).

This conceptualisation provides an answer to the principal research question on how the tensions between the specificity of designs and the distributedness of knowledge and control across the software ecosystem can be resolved. The process of the *generative-integrative mode of developmen*t outlines a scenario where under-specification and constructive ambiguity enable the initial generative combination, yet the post-combination integration efforts are needed to remove the incongruences which emerge from the absence of system-level design principles and the specificity of designs.

7.2 Enfolding literature

This section enfolds the conceptual findings in the light of previous literature (Eisenhardt 1989b). The purpose of this is to sharpen the findings, establish the boundaries of generalisability and examine and evaluate the ways in which the

findings contribute to the literature on digital innovation. This enfolding is presented with reference to the literature reviewed in Chapter 2. This body of literature theorises the dynamics of combination in the context of product architectures, providing thereby an appropriate backdrop for the enfolding.

The theories presented in the previous literature do not fully capture the organising logic of complex digital innovation as established in the empirical and conceptual findings of this research. These differences are discussed in the following order. First, the conceptual findings are discussed with reference to modularity and the modularisation of product systems. This is then followed by the generativity of digital innovation and digitised products in the view of layered modular architectures and complementary architectural frames. Finally, the differences are discussed with reference to complex products and systems literature before concluding with the validity of conceptual findings.

7.2.1 Integrality and modularity

One of the fundamental conceptualisations in the literature on product architectures revolves around integrality and modularity (Ulrich 1995). As discussed in Chapter 2, product architectures are called modular when the functional elements of a product correspond to the components that constitute it, and the interdependencies between components are such that a change in one component would not require corresponding changes in other components. In turn, product architectures are called integral when multiple components in conjunction produce a particular functional element, and a change in one component is likely to trigger changes in others. Most product architectures reside somewhere between the modularity and integrality (Salvador 2007).

In terms of integrality and modularity, contextually bound and embodied chains of transformation are much closer to the integral than the modular end of the spectrum. While it is possible to identify a component that performs a particular transformation, for example, an object recognition component that identifies particular patterns from a stream of images, the behaviour of that particular transformation constitutes an integral part of the overall chain of transformations that produces the behaviour of a system with respect to its environment. While it would be technically possible to replace a pattern

recognition component with relative ease as long as the necessary syntactic interfaces are specified and adhered to, a change in the behaviour of that particular component in terms of its input-output mappings between the images and their respective classifications would alter the behaviour of the whole system. In addition, although under-specification and constructive ambiguity may facilitate the generative combination, implicit but misaligned assumptions may prove detrimental[86] to the reliability and robustness of the system. This integral character is also evident in the generative-integrative mode of development where efforts are made to craft generative combinations into purposeful and dependable compositions. This way, the context-dependent and embodied chains of transformation can be viewed as highly integral.

This leads to the question to what extent the chains of transformation can be modularised? In the end, following Baldwin and Clark (2000), modularisation is a process where the overall functionality of a product is allocated to its constituent components according to specific rules so that the design and manufacturing tasks can be distributed across different teams and organisations (Sanchez & Mahoney 1996). This relies on the assumption that the architect who modularises the functionality over different components is aware of the overall requirements sufficiently well to be able to specify how different components interact with and depend on each other (Baldwin & Clark 2000). Subsequently, designers of particular components would have a limited degree of freedom to carry out the design and implementation in a most feasible way as long as the specifications and design parameters of the overall design are met. By designing degrees of freedom into the parameters of module design, modules can be seen as units of variation (Salvador 2007), which can be then be combined in different ways to produce variety in products (Schilling 2000).

When modularising chains of transformation, probably the first questions to ask would be in which way and according to which logic should a behavioural model produced by a chain of transformations be varied within a particular scheme of modularisation. The approach to modularisation would presumably vary

[86] To provide an example, NASA lost a Mars orbiter in 1999 due to a units of measurement mismatch that prevented the transfer of navigation information between the Mars Climate Orbiter spacecraft team at Lockheed Martin in Denver and the flight team at NASA's Jet Propulsion Laboratory in Pasadena. Lloyd, R. and Writer, C.I.S., 1999. Metric mishap caused loss of NASA orbiter. *CNN Interactive*.

according to the expected behaviours and scope of required variation (Salvador 2007). Furthermore, the modularisation of a pattern recognition system would probably differ from the modularisation of physical embodiments and associated motion planners. Establishing an overall framework of design that would enable the replacement of components to adapt the behavioural model for new tasks and task environments module by module would presumably require careful information engineering to ensure that the patterns of interaction among the computational processes would produce desired behaviour.

Considering that the observed generative-integrative mode of development does not rely on exhaustive a priori specification, modularisation or centralised coordination mechanisms, the top-down approach to the modularisation of behavioural models cannot be elaborated here further. Instead, it is observed that through the generative-integrative mode of development it is possible to develop integral systems from components that originate from heterogeneous sources. Therefore, whereas contextually bound and embodied chains of transformation can be viewed as integral, they cannot be fully understood through the lens of integrality (Ulrich & Eppinger 2012), centralised design agency or modularisation (Baldwin & Clark 2000).

7.2.2 Generative combinations

In the literature on digital innovation, the notion of generativity (a form of diachronic emergence) is used to explain the growth and evolutionary dynamics of the Internet (2008; 2006), digital platform ecosystems (Yoo et al. 2010) and digital infrastructures (Tilson et al. 2010). Whereas the notion of modularisation builds on the assumption of the centralised and coordinated design and decomposition of product architectures (Baldwin & Clark 2000), generativity describes digital innovation as a combinatorial process where novel digital products and assemblages of digital objects and artefacts are created by generating new combinations of existing objects and artefacts (Yoo et al. 2010) which originate from heterogeneous sources in the absence of central coordination (Yoo 2012b).

The notion of generativity upends the assumption of the top-down centralised design and portrays digital innovation through the lenses of emergence and assemblages that emerge from distributed actions of heterogeneous groups of actors in a bottom-up fashion (Yoo 2012b). For this to occur, the digital objects and artefacts to be combined only need to conform to certain common foundations, such as open interfaces and communication protocols as they facilitate the interoperability and combinability of digital objects and artefacts even if they belong to different design hierarchies (Yoo et al. 2010; Clark 1985). In Zittrain's (2008; 2006) work, these common foundations are located at the operating system of a computer and the internet standards and protocols. The former provides a layer of abstraction between the computer hardware and application software, whereas the latter allows for application software and all sorts of digital objects to be transferred over the digital communication networks and infrastructures (Tilson et al. 2010). More broadly, the generativity of digital innovation is postulated to follow from the layered characteristics of digital architectures (Yoo et al. 2010) given that different components can interact with each other through well-specified interfaces, which hide the internal workings and implementation details (Parnas 1972). In this view, as long as interfaces are well-specified, and components produce specified functions, developers can focus on their own areas of work without needing to know much about the internal functioning of other components. Therefore, the generative properties are founded upon the uncoordinated division of labour which is facilitated by the standardised and open platforms and interfaces and common communication protocols, which enable the combination of end-product agnostic components originating from heterogeneous sources. The generative characteristics of digital innovation render knowledge and control highly distributed among different actors and organisations (Yoo et al. 2010).

Based on the empirical findings, the mode of system development was conceptualised as generative-integrative. This characterisation highlights the fact that developers may choose from the components which originate from heterogeneous sources and then combine them using the ROS communication system as it provides common message types and methods of communication for establishing the interconnections between various components. In this view, the empirical findings give a clear demonstration of the generativity that rests

upon the common communication methods and protocols. However, considering that the ROS-based software is a cluster of distributed computational processes, ROS is not generative in the sense of the traditional computer operating system with its central platform and stable interfaces. There is no central and stable abstraction layer in the form of a single operating system that would provide a set of interfaces upon which further functionality could be developed.

Furthermore, as discussed earlier, simply joining components together rarely produces desired behaviour. Contextually bound and embodied chains of transformation are integral and distributed, and efforts are needed to integrate the combinations of components into dependable compositions. This requires knowledge not only of the components but also of their interactions and interdependencies.

Whereas the empirical findings observed in the context of ROS could be partially characterised as generative, it can also be concluded that the notion of generativity (Yoo et al. 2010) on its own would not be sufficient to explain the empirical findings.

7.2.3 Layered modular architecture

The notion of the layered modular architecture is a combination of the modular architecture of products (Ulrich & Eppinger 2012) and the layered architecture of digital technologies (Yoo et al. 2010). The purpose of this combination is to bring forward the way in which the architectures of modern digitised products incorporate the characteristics of both the top-down driven modularisation and the bottom-up driven generative combination (Yoo et al. 2010). The layered modular architecture conceptualises the architecture of digitised products as a layered stack. The device layer resides at the bottom of the stack. It consists of the physical hardware and computing machinery and includes also the operating system that provides a logical layer of abstraction that modulates interaction between the hardware, computing machinery and upper layers of the software stack. On the top of the device layer resides the network layer. The network layer consists of the physical transport media (e.g. antenna, cable) and the logical transmission protocol (e.g. TCP/IP), and it is used to establish

interconnections with other computers and digitised products. On the top of the network layer is the service layer where application software and their respective functionality are located. Above the service layer, on the top of the stack, is the contents layer that holds the data contents. The device layer can take many forms along the spectrum from a traditional manufactured product to a desktop computer (Yoo et al. 2010). Depending on where a digitised product is located along this spectrum, the extent to which it is open to ongoing change and generative combinations varies. Whereas the hardware and computing machinery residing on the device layer usually receive their final configuration and form during the manufacturing process, the layered stack of software that resides on the top of it is more amenable to change throughout the lifecycle of a product (Yoo et al. 2010). The device layer provides also a layer of abstraction in the form of an operating system and application programming interfaces, making it possible to alter the functionality and purpose of a product later by modifying the software at the higher levels of the stack. If a product manufacturer engages in open innovation by opening up interfaces (Eaton et al. 2015), a digitised product can become a platform that stimulates generative innovation by allowing broader audiences to take part in the development of new functionalities and services.

With reference to embodied chains of transformation, the notion of layered modular architecture can be viewed in two different ways, depending at which level of a design hierarchy the notion is applied. In the view of the constituent elements of robot systems, the physical embodiments are divided into three categories: sensors, actuators and hardware platforms. On one hand, sensors and actuators can be viewed as components having the layered modular architecture at the level of individual components. On the other hand, also a hardware platform that encapsulates sensors, actuators and other hardware components into one unified product can be seen as a platform of the layered modular architecture, as long as there is a clear layer of abstraction that provides a set of application programming interfaces for dealing with sensory data, action commands, hardware parameters as well as for monitoring and diagnostics.

Some hardware platforms offer well-specified functionality and interfaces, which have been made compatible with ROS. Commercially manufactured quadrotors provide a good example of this type of hardware. The manufacturer designs and produces a quadrotor that is a fully functional product on its own but at the same time serves as a platform with open interfaces. This allows for broader audiences to take part in innovation. The use of smartphones as hardware platforms provides a similar example. Smartphone manufacturers provide open interfaces to access various sensory readings and to control screen and speakers among other functions. While these types of hardware platforms with open interfaces can be viewed as stable and central core components, the chains of transformation which produce the models of behavioural can also be viewed as applications. If a decision is made to draw boundaries around a particular distributed cluster of computation and to refer to it as an application which resides at the service layer, then it would be possible conclude that a combination of a hardware platform (device layer) and chains of transformation (service layer) are compatible with the notion of the layered modular architecture.

On the other hand, when embodied chains of transformation are constructed by bringing together a distributed set of sensors, actuators and processes of transformation, there is no particular platform that could be considered on its own to serve as a stable foundation upon which the upper layers facilitating the generative innovation reside. Instead, the embodied chains of transformation comprise a heterogeneous and intertwined group of physical embodiments, brought together and controlled by the systems of communication and transformation. In this case, the absence of foundational and stable platforms and associated layers of abstraction is not compatible with the underlying assumptions of the layered modular architecture.

Therefore, the notion of layered modularity (Yoo et al. 2010) cannot be considered as generally applicable in the context of embodied of chains of transformation and the generative-integrative mode of development. Instead, the applicability of the notion is contingent on the level of platformisation and encapsulation at different layers of the device and software stack and the associated stability of interfaces.

7.2.4 Complementary architectural frames

Another conceptualisation of digitised products introduces the notion of complementary architectural frames (Henfridsson et al. 2014). This conceptualisation proposes that the architectures of digitised products can simultaneously be seen as hierarchies of parts and networks of patterns. The hierarchy of parts view refers to the physical architecture of a product, which is expected to be fully designed and specified before it can be transferred to manufacturing. The network of patterns, in contrast, refers to a set of interconnected functionalities that can be implemented and changed later using digital means. However, while the patterns serve as placeholders that facilitate change at a later point in time, the product architect who performs the functional decomposition and allocation of functionalities over both the parts and patterns is expected to have an overview of the overall functionality and its expected development paths as well as the limits of variation in a given product system (Salvador 2007). Therefore, the use of the complementary frames shares the assumptions of top-down design and well-specified interfaces with the notion of modularisation (Baldwin & Clark 2000). The interdependency among the complementary architectural frames resonates also with the intertwined design hierarchies of inclusion and control (Murmann & Frenken 2006).

This conceptualisation has similarities to the notion of embodied chains of transformation. To begin, the prominent role of physical embodiments in product design and architectures is acknowledged as the material substrate upon which the digital functionalities are implemented. Also, the emphasis on networks highlights the distributed character of computation as functionalities are distributed across a number of digital components. However, the notion of patterns, as it is used and illustrated in the context of complementary frames (Henfridsson et al. 2014), is not entirely applicable in the view of chains of transformation as it is primarily used to signal the differing speeds of design cycles between the physical components and digitally implemented functionalities. While the notion indicates the procrastinated binding of the digital part (Yoo 2012a), it puts little emphasis on the specificity of designs and the physical groundedness of computation that characterise embodied chains of transformation. In addition, the assumption of top-down design is not entirely compatible with the generative-integrative mode of development, which

proceeds from the generative combination towards the increasing levels of integration.

7.2.5 Complex products and systems

The innovation literature on complex products and systems argues that the development of complex systems requires detailed knowledge of different components and their interactions (Prencipe 2000). In this view, complex products and systems are conceptualised as compositions of two design hierarchies that are intertwined and interdependent (Lee & Berente 2012; Murmann & Frenken 2006). These two hierarchies are the hierarchy of inclusion and the hierarchy of control. The hierarchy of inclusion refers to the hierarchical and nested organisation of parts which constitute the physical embodiment of a product or system. In turn, the hierarchy of control refers to the parts and functional logic that control the operation and behaviour of that embodiment. The architects that design control systems are expected possess detailed knowledge not only of different components but also of their various interactions, interdependencies and behavioural dynamics (Prencipe 2000). Building a control system is predicated on knowing what is being controlled, for what purpose and under what boundary conditions. Therefore, building a complex system requires often a significant amount of contextual knowledge as well as modular and architectural innovation (Henderson & Clark 1990).

In addition, complex products and systems are also occasionally differentiated on the grounds of the market conditions that separate them from the industrial mass-market products and associated innovation dynamics (Miller et al. 1995; Hobday 1998), this being particularly the case when various stakeholders collaborate to produce unique architectural designs and technological compositions that transcend industry-wide dominant designs (Abernathy & Utterback 1978) and design hierarchies (Clark 1985). Flight simulators and nuclear power plants (Miller et al. 1995; Hobday 1998) provide a case in point.

The conceptual findings of this research align to an extent with the literature on complex systems and products. The concept of contextually bound and embodied chains of transformation indicates the presence of the hierarchies of inclusion (embodiments) and control (transformation). It also emphasises that

chains of transformation are conditioned not only by the embodiments but also by the contextual bindings, showing that the development of chains of transformation entangles knowledge around the operational environment, physical embodiments as well as various digital methods and technologies. This brings forward the specificity and integrality of designs that spread throughout the chains of transformation, indicating the need for both component and architecture level knowledge and innovation (Henderson & Clark 1990). However, while this chains of transformation view is compatible with the dual-view of the design hierarchies of inclusion and control, the chains of transformation view does not assume a hierarchical control system but accommodates parallel and multi-agent control architectures as well.

In addition to the emphasis on the specificity of designs, the complex systems literature also highlights the need for coordinated collaboration. While this is present in the integrative part of the generative-integrative mode of development, the generative part is not entirely compatible with the underlying assumptions concerning the depth and breadth of knowledge and control (Prencipe 2000). This assumption maintains that the organisations which engage in complex innovation are expected to possess detailed knowledge at the level of product architectures and constituent components. However, the empirical findings of this research show that knowledge and control can be highly distributed across a heterogeneous body of users and contributors that operate in an uncoordinated manner.

While the innovation literature on complex systems and products captures much of the underlying dynamics of complex digital innovation, it is not particularly well-suited to explain the distributedness of knowledge and control or the unfolding of the generative-integrative mode of development.

7.2.6 Summary

The evaluation of the conceptual findings in the light of the reviewed literature demonstrates that the current conceptualisations do not fully capture the organising logic of complex digital innovation as observed in the context of robots and autonomous systems. Some of the existing conceptualisations are able to explain some aspects of the empirical findings, but, overall, they do not

provide a conceptualisation that would provide a sufficient explanation of the observed dynamics and practices of innovation in the context of complex digital innovation.

Therefore, it can be concluded that the proposed concepts bring forward salient characteristics of complex digital innovation in a holistic manner and offer useful additions to the conceptual toolbox of digital innovation research. The structural-functional conceptualisation presents the structural and functional factors that underlie complex digital innovation, whereas the generative-integrative mode of systems development that builds on underspecification and constructive ambiguity sheds light on how the tensions between the specificity of designs and the distributedness of knowledge and control are resolved.

7.3 Application of the proposed concepts

This section discusses the applicability of the proposed concepts outside the empirical setting where they were first developed while also discussing briefly how they could be developed further. The concept of chains of transformation is first discussed and forwarded as a more tractable conceptualisation not only to analyse robots and autonomous systems but also other artificial intelligent and cybernetic systems that can be considered as complex digital innovation. The view provides a highly generalisable lens to examine functional and organisational characteristics of complex digital innovation providing a pathway towards a more fine-grained examination of digital and computational "value chains" that produce value through the processes of transformation. Second, the generative-integrative mode of development is further elaborated in the view of simulation studies and the levels of conceptual interoperability model. This provides a conceptual explanation of the observed unfolding of the generative-integrative mode of development and outlines fruitful avenues for further research.

7.3.1 Chains of transformation

The concept of contextually bound and embodied chains of transformation can be extended and applied to the adjacent areas of complex digital innovation such as the systems of artificial intelligence and cybernetics. Viewing these

domains of innovation through the structural-functional lens could provide a way to construct research questions in a way that would take different organisational and functional characteristics into account in a more holistic manner. Viewing such systems as contextually bound and embodied chains of transformation would bring forward the distributed and transformative character of computation while highlighting the context-dependence of purposeful transformations and the physical groundedness of computation.

Many sophisticated special-purpose computing systems and computational processes that carry out transformations fall into a category which is often defined as artificial intelligence (Russell & Norvig 2010). While the labelling of sophisticated computation as artificial intelligence might be appropriate in some occasions, its meaning is not particularly precise. Furthermore, there are multiple fields of advanced computation which can be considered as subfields of artificial intelligence. These subfields tend to focus on different aspects of computation, developing and working on algorithms, techniques, methods and computational strategies that are capable of carrying out computations that solve some particular types of problems, for example to recognise pertinent patterns in sensory readings, to fuse a variety of recognised patterns or to control motion over time and in real-time to provide few examples. In this light, the common aspect is that they are all created and designed to perform some particular transformations and tp serve specific purposes or functions.

The application of the notion of chains of transformation would help pose more detailed questions concerning the functional, structural and organisational aspects of robots and autonomous systems and artificial intelligence in their various guises. This would allow us to ask questions such as what is being transformed and on what basis? More detailed questions could revolve around why something is being transformed, are different transformations compatible with each other, or what constitutes a computational “value chain” and who captures the value from transformations and how? Questions could also be asked on who controls transformations and on what grounds some transformation is better than others. Focusing on the computation and transformations as well as the purposes they serve instead of robotness, autonomy and artificial intelligence would remove the unnecessary mysticism of

the debate and could help direct attention and research questions to the matters that are more grounded and empirically tractable.

Moreover, as robots and autonomous systems as chains of transformation are expected to operate with limited human intervention towards a given goal based on the environmental inputs and related control and feedback mechanisms, chains of transformation can be viewed as constitutive elements of cybernetic systems. While cybernetics focuses the role and dynamics of information and feedback in the control of systemic behaviour, it does not tell much about the internal arrangements or organisation of such systems or how this may influence innovation practices. The notion of chains of transformation can be viewed as more descriptive in this view while being simultaneously consistent with the idea of cybernetic. In the end, the cybernetic systems and feedback loops are contextually bound and composed of embodied and interconnected transformations, which in conjunction transform sensory inputs into systemic actions and behaviour.

Finally, the notion of chains of transformation directs attention to the importance of systems integration. To achieve a desired systemic behaviour, various computational processes must be carefully arranged and integrated. Simply joining components generatively together does not necessarily lead to a desired system-level behaviour; the interoperability of components does not imply that the transformations in conjunction are semantically meaningful. Chains of transformations are like production lines in factories where stages of manufacturing have to be carefully arranged and integrated in order to transform raw materials into high-quality end products (Arthur 2009).

7.3.2 Generative-integrative mode of development

The proposed concept of the generative-integrative mode of development offers an insight on how the tensions between the specificity of designs and the distributedness of knowledge and control can and cannot be resolved. As this concept seems to be not fully compatible in the view of the reviewed theories that seek to explain the organising logic of innovation from the perspective of product architectures and combination, it requires further elaboration. To this end, this section refers to the lessons from simulation studies and elaborates on

the role of semantic compatibility in the development of chains of transformation.

The generative-integrative mode of development blends the uncoordinated and distributed innovation practices with the specific and integral design requirements. The initial combination is facilitated by the under-specification of interconnections and constructive ambiguity, and the subsequent iteration cycles seek to increase the level of integration among different components and with respect to the task and task environment.

The development of chains of transformation means of joining a series of transformations (computations) over the connection that transfers symbolic representations (data). This is a brittle and context-dependent endeavour. Each symbolic representation is an abstraction that captures some aspects of the state of affairs while ignoring others, and each transformation receives its meaning by converting the abstraction that serves as an input into some subsequent abstract output, a representation that again serves as an input for the next process of transformation (Floridi 2013; Floridi 2008; Bekey 2005). In principle and in practice, a single inappropriate transformation somewhere along chains of transformation, such as a misclassification of an object in the image, may result as a wrong or outright harmful outcome.

The difficulty of combining a variety of abstractions and transformations is well known in simulation studies (Tolk et al. 2007). Simulations, which are also used as tools in the development of robots and autonomous systems, are often used to model and evaluate the behaviour and dynamics of complex systems and processes over time, especially scenarios that are not necessarily analysable using linear or analytical methods. While the components used in simulations are in principle transferable and reusable due to their digital characteristics, they often are much less so in practice. The spectrum of implicit assumptions and context specificity that are embedded in architectural arrangements, abstractions and transformations often act as a barrier for composability and reuse (Spiegel et al. 2005; Tolk et al. 2007).

An efficient reuse and combination of simulation components would require a comprehensive method for identifying and validating critical constraints, although the identification and conservation of such constraints is challenging and may remain beyond human capabilities (Spiegel et al. 2005). While both the component-based software engineering and the composition of simulation models require syntactic and semantic interoperability (Bartholet et al. 2004), the composition of simulation models can be viewed more challenging considering that the simulation models are often of large-scale and expected to command a high degree of realism and semantic validity. Simulations, as any models, are bound and constrained by their design and underlying assumptions, and a failure to appreciate the difference between an abstract computational model of simulation and the underlying phenomenon it seeks to represent may lead to misjudged decisions and unwanted organisational outcomes (Bailey et al. 2012).

As previously discussed, software components that originate from distributed and heterogeneous sources can be interconnected to each other as long as they adhere to the same syntactic structure that consists of a message type and method of connection, yet the ability to exchange a message does not guarantee that the established interconnection and resulting systemic behaviour would be meaningful. This tension between the syntactic interconnectedness and semantic interoperability is captured by Tolk, Diallo and Turnitsa (2007) who argue that the meaningful interoperability in simulations requires much more than technical layers of interoperability:

"The challenge is not to exchange data between the system: the technical side is sufficiently dealt with by interoperability standards. The problem is that the concepts of the underlying models – or the implemented world view captured in the model – need to be aligned as well" (Tolk et al. 2007)

To conceptualise the different levels of interoperability, Tolk, Diallo and Turnitsa (2007) present the Levels of Conceptual Interoperability Model (LCIM) that describes six different levels of interoperability. These levels, the technical, syntactic, pragmatic, dynamic and conceptual layers of interoperation are presented in Figure 40. The technical level indicates the existence of communication protocols for exchanging data. The syntactic level refers to a common data format to exchange data, whereas the semantic level refers to the

shared meaning of exchanged data. The pragmatic refers to the level of interoperability where components or systems operating in conjunction are aware of the methods and procedures that each other are employing and are aware of the overall context of their application. Furthermore, the level of dynamic interoperability is reached when the system, as it operates over time, can change its state and adapt to new assumptions and constraints that have implications on the exchange of data among the different parts of the system. Finally, the highest level of interoperability is reached when *"the assumptions and constraints of the meaningful abstraction of reality – are aligned"* (Tolk et al. 2007 p. 67).

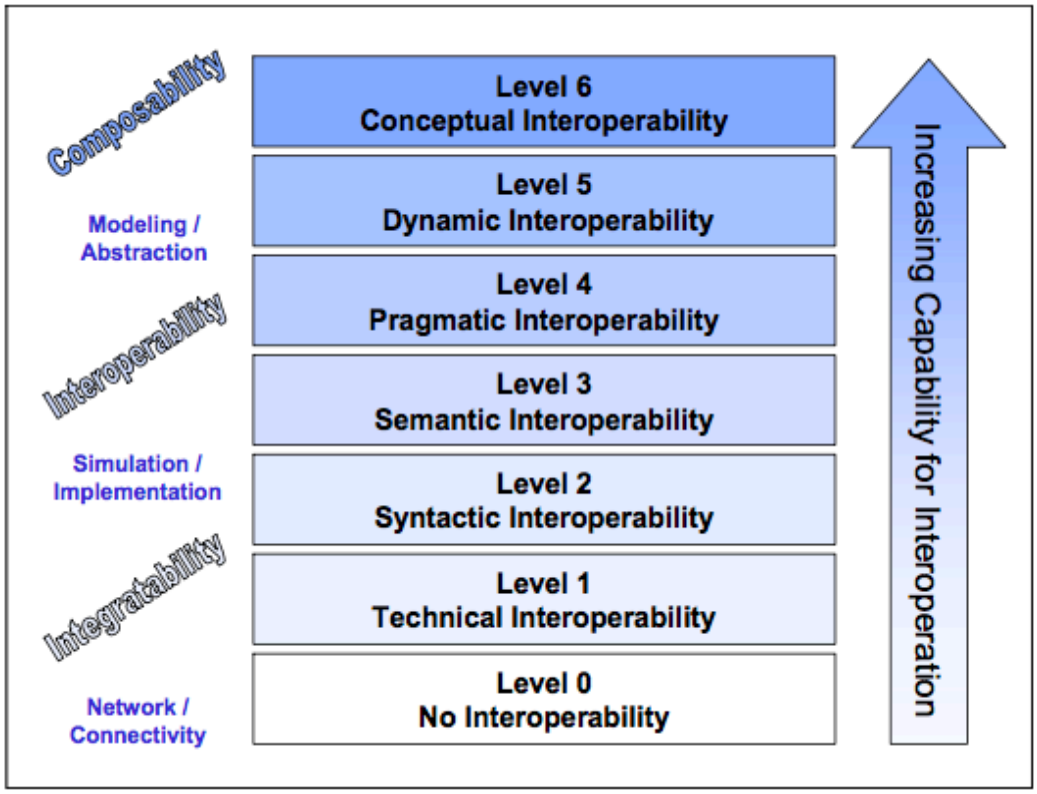

Figure 40: The Levels of Conceptual Interoperability Model (LCIM) (Tolk et al. 2007)[87]

A successful description of different levels of interoperability requires well-specified ontological schemes, yet it is uncertain to what extent the construction of any general and overarching schemes is achievable or feasible without losing the fine print of contextual reality. Many researchers of complex systems, such

[87] Republished from the Journal of Systemics, Cybernetics and Informatics, Applying the Levels of Conceptual Interoperability Model in Support of Integratability, Interoperability, and Composability for System-of-Systems Engineering, by Andreas Tolk, Saikou Y. Diallo, Charles D. Turnitsa, 2007, 5(5), page 66, with permission.

as social organisations, are aware that their conceptualisations can capture only a fraction of reality at any given moment and often lack predictive capacities.

Against this backdrop, the generative-integrative mode of development appears entirely plausible. In the light of the LCIM model, the lower levels of interoperability provide a possibility for generative combination, whereas climbing up to the higher levels of interoperability requires additional integration efforts. In the context of the ROS communication system, the interoperability occurs at the technical level (interconnection) and the syntactic level (message type), whereas the level of semantic interoperability tends to take place at the realm of common conventions that revolve around shared understandings of the conventional use and purpose of specific message types and methods of communication. While this makes it possible to connect different components and frameworks together, a developer is left with a task of constructing the higher levels of pragmatic, dynamic and conceptual interoperability between transformations in order to achieve an appropriate level of behavioural couplings that is congruent with the physical embodiment, task and task environment.

The LCIM model and traversing upwards along the layers of interoperability and abstraction appear as a potential conceptual explanation concerning the under-specification of interconnections and constructive ambiguity. In this light, the differing opinions and interpretations of ROS and the perceived transferability of software packages could possibly be explained in terms of expectations regarding the levels of interoperability. While the LCIM model offers a path to explanation, additional research and conceptualisation are needed to establish in a more detailed manner how these different levels of semantic interpretability unfold in complex digital innovation. As digital representations and chains of transformation become more convoluted, growing in size and complexity, the interoperability can be expected to affect the distribution of software and other digital objects and artefacts as well as have implications to the practices of systems development. This line of inquiry may provide a fruitful avenue of research for those who are interested in explaining and theorising the logic of combination in the context of complex digital innovation.

7.4 Summary

This chapter restated and discussed the conceptual findings of this research. These findings conceptualise the structural-functional characteristics and organising logic of complex digital innovation in the view of product architectures and combination.

To begin, robots and autonomous systems were conceptualised as contextually bound and embodied chains of transformation. This is to highlight that these systems render their behaviour through the interaction with the surrounding environment, are physically embodied and comprise interconnected computational processes which transform sensory inputs into situated actions. The direct coupling with the surrounding environment and the qualitative difference between the inputs and outputs differentiates robot systems from the traditional information and communication systems that mediate messages and digital objects that are created and consumed by people. This shift marks the expansion of the focus of digital innovation research from the information systems to transformation systems. This conceptualisation provides a foundation for further research towards a more fine-grained examination of computational "value chains" by bringing forward simple questions such as what is being transformed, how are transformations joined together or who controls transformations and related abstractions. Moreover, perhaps, more importantly, it would also possible to ask who decides which of ones the transformations among the infinite variety of possible transformations are the ones that will be deemed fit, just and appropriate and on what grounds that is to take place.

Then, the generative-integrative mode of systems development was presented as an approach that iteratively resolves the tensions between the specificity of designs and the distributedness of knowledge and control. This can be viewed as a mode of development that gradually seeks to increase the level of conceptual interoperability among the components that constitute a system. The generative-integrative mode can be described as an approach where an initial generative combination of existing components provides the starting point for further systems integration efforts. While the first combination may build upon the under-specification of interfaces and constructive ambiguity, the subsequent

integration efforts seek to establish a higher degree of interoperability among the components as the initial combination of components is gradually coevolved and crafted into a composition that is expected to produce meaningful, robust and dependable computer-controlled behaviour.

8 Conclusions

This is the final chapter that concludes this thesis. The chapter begins by presenting an overview of the work and then provides a summary of key findings. Subsequently, the findings are discussed to bring forward the conceptual contributions developed in the previous chapters. This is followed by the examination of the validity and limitations of this work, after which potential avenues for future research are briefly presented.

8.1 Overview of the thesis and summary of the findings

The research presented in this thesis focuses on the organising logic of complex digital innovation in the context of robots and autonomous systems from the point of view of product architectures and combination. To date, this area of research has received relatively little attention among digital innovation, information systems and management researchers. Considering that robots and autonomous systems in their different forms and functions are expected to play an increasing role in the future organisational, social and economic arrangements, the findings presented in this research are expected to be of interest to both academia and industry as they describe and conceptualise the organising logic of the systems that intertwine the physical, digital and autonomous aspects of technology.

8.1.1 Background and research questions

Robots and autonomous systems can be viewed as a type of innovation that is both complex and digital – they can be viewed either through the lens of digital innovation or through the lens of complex products and systems innovation. However, these two streams of innovation research draw their lessons from different empirical settings and present incompatible logic and principles of combination in the view of product architectures as presented and discussed in the literature review in Chapter 2.

Product architectures are often discussed in terms of integrality or modularity depending on the way in which functional elements are assigned to physical components (Ulrich 1995). In modular architectures, the functional elements of

a product correspond to the components that constitute it, so that by changing a component, the corresponding functionality changes. Integrality, in turn, refers to the situation where components produce a particular functionality in conjunction so that functionalities cannot be changed in a piecemeal manner by changing some particular components in isolation. Modular product architectures are produced through the process of modularisation during which the overall design of a product is decomposed into its constituent components in a top-down manner (Salvador 2007; Baldwin & Clark 2000). Thereby, the knowledge of the overall architecture is a precondition for successful modularisation if it is to facilitate the separability and combinability of different components and modules (Schilling 2000).

The modularity of digital components and assemblages is said to differ from that of physical products and components. Digital product architectures are often represented as layered stacks (Yoo et al. 2010), which are modularised so that different layers and components perform specific functions and services which are accessed through well-specified interfaces and communication protocols. What happens inside a component can be hidden and abstracted away from its users (Parnas 1972). Therefore, as long as the functioning of a component is well-specified and the interface for accessing the component remains stable, it can be reused and its internal workings can be changed. By adhering to specifications, open standards and common protocols, software developers can focus on their particular areas of work with little regard to what is abstracted away and hidden from them. This encapsulation enables generativity as new ensembles can be created by combining components from heterogeneous sources in a bottom-up manner (Yoo 2012b; Zittrain 2008; Zittrain 2006). This way, whereas modularity assumes the knowledge and control of an overall product architecture for modularisation efforts to succeed, the generativity of digital innovation does away with the centralised design agency rendering the knowledge and control over technologies and innovation trajectories highly distributed (Yoo et al. 2010).

The literature on complex systems and products also casts doubt to what extent straightforward modularisation is feasible, but it does so on very different grounds (Prencipe 2000). The architectures of complex systems are

conceptualised as consisting of the hierarchies of inclusion and control (Murmann & Frenken 2006; Lee & Berente 2012). The hierarchy of inclusion refers to the physical components and parts that form the embodiment, and the hierarchy of control refers to the systems that control the embodiment. The elements that comprise these two analytically separate hierarchies are highly intertwined and interdependent, rendering such architectures with a high degree of specificity, integrality and complexity. It is argued that the companies which engage in the complex products and systems innovation are expected to possess detailed knowledge not only of the interdependent design hierarchies and also of the functioning of the components that constitute them (Prencipe 2000). The high degree of interdependence among components makes the efforts of modularisation challenging. Furthermore, as the components that comprise the hierarchy of control often rely on digital computation, they can be referred to as digital control systems (Lee & Berente 2012).

To problematise, in the context of complex digital innovation, should the organising logic of innovation be viewed through the lens of generative combinations or perhaps through the lens of complex systems and specificity?

With reference to this problematisation, the principal research question and two operative research questions were formulated to guide the empirical investigation. The principal research question is stated as follows:

How can the tension between the specificity of designs and the distributedness of knowledge and control be resolved in the development of complex and digitised products?

The principal question needs to be restated in operative terms to make the process of data collection and analysis more tractable (Hintikka 1999). To this end, operative research questions were constructed upon tentative a priori concepts based on Herbert Simon's (1996; 1962) theory of hierarchy that conceptualises the structures of complex systems as nested and recursive structures that are nearly-decomposable.

From this starting point, the efforts were first concentrated on the identification of the instances of subsystems irrespective if they belonged to the hierarchies of

inclusion or control or at which hierarchical level they resided. The first operative research question is expressed as follows:

What are the typical instances and characteristics of subsystems, if any?

Once the instances of subsystems were established, the focus of research shifted from the subsystems to their combinations, with an emphasis to explore how subsystems are connected at and across different levels of design hierarchies. This is expressed through the second operative research question as follows:

What are the typical instances and characteristics of combinations, if any?

Examining the subsystems and their respective combinations was expected to increase the understanding on how tensions between the specificity of designs and distributedness of knowledge and control are resolved in the development of complex and digitised products.

8.1.2 Approach to research

The Robot Operating System (ROS) (Quigley et al. 2009) was a selected as a case to study after the pilot study phase, and the research was designed as an embedded case study (Yin 2009). ROS was chosen as it was considered to provide a holistic and representative view of the development of robots and autonomous systems. An extensive research database was constructed primarily from the publicly available documentary evidence that covered ROS and the ROS community. This was complemented by the field notes and observations that were gathered during different workshops and events.

The research database was processed to serve two different purposes, to develop a case description and to carry out the thematic analysis that would provide an answer to the principal research question. The analysis (Silverman 2015) unfolded over five different although partially overlapping phases, familiarisation, open coding, categorisation, thematisation and conceptualisation. The first phase focused on getting familiar with the field of robotics. The second phase (open coding) labelled data using the operative research questions and their conceptual underpinnings as sensitising devices

(Klein & Myers 1999). The main outcome of these two phases is the case description presented in Chapter 5 which is one of the empirical findings of this research.

Subsequently, the third phase (categorisation) examined the labelled data in more detail to locate recurring themes that would warrant the categorisation in the view of research objectives. This iterative and interrogative process (Hintikka 1999) produced 15 categories, which were further abstracted to six themes during the fourth phase (thematisation). Then, finally, the fifth phase (conceptualisation) evaluated and established the relationships among the themes and categories. The conceptual propositions that were developed based on the themes and categories constitute the theoretical contribution of this research. Finally, the primary research question was answered with reference to the theoretical propositions.

The contributions of this research can be divided along the lines of empirical findings and conceptual propositions. The case description, themes and categories constitute the empirical findings, whereas the theoretical contributions comprise the two proposed conceptualisations. The empirical findings and conceptual propositions are briefly presented in the subsequent sections.

8.1.3 Empirical findings

The case description, themes and categories that were developed during the process of thematic analysis form the main empirical findings of this research.

The case description presents ROS as a software development framework and open-source community. ROS brings roboticists together and provides them with building blocks, methods and tools that are needed to develop robots and autonomous systems as distributed computers. The case description describes how the two research projects started at Stanford University around 2005 evolved under the auspices of different organisations into a vibrant open-source community while outlining the plans and (un)planned outcomes. This provides unique insight into the domain of complex digital innovation that has so far received little scholarly attention.

After developing the case description, the research database was analysed further to identify recurring patterns, categories and themes. The analysis of resulted in 15 categories, which were further abstracted to six themes: robot systems (1), physical embodiments (2), communication systems (3), transformation systems (4), visualisation and testing systems (5) and the ROS community and software development (6). These themes and their respective categories are outlined below.

To begin, the theme of robot systems (1) consists of two categories that are research robots and productive applications, indicating the overall purposes of robot systems. As a composition, a robot system comprises three different themes, which are physical embodiments and communication and transformation systems. These three themes are further divided into categories. The theme of physical embodiments (2) consists of three categories that are sensors, actuators and hardware platforms. They provide a robot system with the means to interact with its surrounding environment. The theme of communication systems (3) deals with messaging and coordination systems and connectors, and their primary purpose is to facilitate the setting up of a robot system as distributed computation. Whereas the messaging system handles the run-time messaging among distributed computational processes, the coordination systems provide the functionality for managing, coordinating and monitoring the operation of distributed computation. The theme of transformation systems (4) consists of the systems that carry out transformations over coordinate frames and other representations of data. The coordinate transformations perform transformations between different geometric coordinate frames over time, whereas the representational transformations, in general, perform qualitative, gradual and stepwise transformations which convert the inputs of one kind to the outputs of another kind, such as the measured patterns of light to action commands to motors. The theme of visualisation and testing systems (5) consists of systems for visualisation, simulation and test data management. The developers of robot systems rely heavily on visualisation and simulation tools in the development of behavioural models. Moreover, as physically embodied processes of distributed computation are often convoluted and difficult to reason, the examination and evaluation of behavioural models tend to be experimental while relying heavily

on the tools of visualisation and simulation. The theme of the ROS community and software development (6) brings forward knowledge transfer efforts and the supporting infrastructure and tools that are designed to facilitate software and technology transfer and reuse in the community.

This categorisation and thematisation revealed the heterogeneity of software packages, subsystems and components that are made available through the ROS infrastructure. The analysis also revealed the absence of platforms, especially if the concept of a platform is understood to refer to a stable core onto which peripherals are connected via interfaces that are specific to a certain design hierarchy (Baldwin & Clark 2000; Baldwin & Woodard 2008). Instead, what is available is a multitude of frameworks and components that are distributed as software packages that serve as building blocks from which robot systems can be constructed. Therefore, while ROS provides the communication system and common conventions around which the community revolves, it is not a platform in the sense as an ordinary operating system would be. In addition, whereas certain central frameworks, such as the ROS communication system and the coordinate transformation systems cater to the needs that are common across different robot systems, the systems and components that perform representational transformations show a great degree of variety. This suggests that the communication systems that perform replication of messages and the coordinate transformation systems that perform transformations across geometric frames of reference are more generalisable than the systems which perform the representational transformations between the qualitatively different inputs and outputs and produce the behaviour of a robot system.

8.1.4 Contributions to literature

This section presents the contributions with reference to the three target areas of contribution as outlined in the introductory chapter. The three areas of contributions are the description and illustration of the empirical domain of robots and autonomous systems, the characterisation of the constitutive elements and organising logic of complex digital innovation and the exposition of the dynamics of innovation at the boundary of generative combination and systems integration.

The case description presented in Chapter 5 provides the first contribution. It provides an overview of ROS, which is a widely used software development framework in the development of robots and autonomous systems. The chapter describes the empirical domain and sheds light on the challenges related to the development of robots and autonomous systems. To date, this framework and community have escaped the innovation and management researchers' attention, and to the author's knowledge, no overall account of the history of ROS has been documented elsewhere. However, ROS is widely used in the field of robotics and it is expected to play a prominent role also in the future. Therefore, the case description opens the door to the emerging domain of complex digital innovation and further investigations.

Second, the conceptualisation of robots and autonomous systems as *contextually bound and embodied chains of transformation* contributes to the literature by conceptualising the salient characteristics that influence the dynamics of combination in the view of product architectures and combination. The purpose of this conceptualisation is to bring forward the structural and functional characteristics that differentiate complex digital innovation from other forms of digital innovation. To differentiate, robots and autonomous systems interact directly with their surrounding environment, are physically embodied and consist of interconnected computational processes which transform between the qualitative different sensory inputs to situated actions. The direct interactional and behavioural coupling between a machine and its environment shifts the locale of interaction and associated sense and decision making processes when compared to the information and communication technologies, which replicate human communications over time and place.

This behavioural, interactive and distributed character of computation bears implications to innovation practices. The shift of focus from transferring information between human actors to the distributed computation which produces goal-directed and context-dependent behaviour poses novel challenges. These challenges are not only related to the reusability and transferability of software, but also to the system design and development practices as well as to the verification of their functionality. Conceptualising the structural and functional aspects of complex digital innovation as contextually

bound and embodied chains of transformation allows us to pose a range of questions regarding the computational "value chains", for example by bringing forward simple questions such as what is being transformed, how to control transformations and related abstractions, how they are brought together and who controls transformations and computer-controlled behaviour in general, among many others.

Third, conceptualising the process of development of chains of transformation as the *generative-integrative mode of development* contributes to the literature on digital innovation by exposing a way how the tensions between the specificity of designs and the distributedness of knowledge and control are resolved. This mode of development can be characterised as a process that begins by combining the first version of a system from components that originate from heterogeneous sources. While this generative combination produces the first version, it cannot be considered as a complete and finished product. Instead, the first version provides a starting point for further systems development and integration efforts, with reference to which the necessary contextual and embodied experimentation and adjustment of behavioural models can proceed. Moreover, as the generative combination builds upon the under-specification of interconnections and constructive ambiguity, additional integration efforts are then carried out to remove the incongruences that emerge from the tension between the specificity of designs and under-specification that is bound to occur in the absence of central design agency and agreed design principles. During the integration phase, the initial combination is gradually crafted into a well-functioning composition that produces the desired behaviour with respect to a task and task environment.

The two proposed conceptualisations in conjunction depict complex digital innovation as a multidimensional and convoluted endeavour and shed light on the organising logic of complex digital innovation. Based on these findings, it also appears that the commonly used references of traditional software engineering, platforms and application stores may not be entirely representative and analogues lenses for studying and explaining complex digital innovation. In addition, the proposed concepts could be complemented and further developed by incorporating and testing the theories that have been developed in the field

of simulation studies. Simulation modellers have studied challenges related to distributed computation and conceptualised different levels of specification, semantic information and interoperability that are needed in the development of computational models of systemic behaviour.

During the course of research, alternative conceptualisations of product architectures and different logics of combination were reflected in the view of empirical findings. The examination revealed that the existing conceptualisations were not able to fully capture the dynamics of innovation observed in the ROS ecosystem. For example, while contextually bound and embodied chains of transformation can be characterised as highly integral (Ulrich 1995) and complex as they consist of the interdependent and intertwined hierarchies of inclusion and control (Murmann & Frenken 2006), these conceptualisations are not entirely applicable as they assume centralised design agency and do not expose the interlinked and stepwise transformations that produce the behavioural models of a robot system. Similarly, while the generative-integrative mode of development aligns partially with the notion of generativity (Zittrain 2008; Zittrain 2006), generativity does not take into account the subsequent phases of development, which iteratively seek to increase the level of integration among the components of a system and with respect to tasks and task environments. Furthermore, while the concept of layered modular architecture (Yoo et al. 2010) is able to explain some of the observations, it does not capture the overall organising logic of complex digital innovation in this empirical context. The layered modularity assumes the presence of a foundational platform and stable interfaces, whereas the development of embodied and distributed chains of transformation is characterised by the absence of platforms.

It can be concluded that the proposed concepts provide a novel contribution to the literature on digital innovation that focuses on product architectures and the related organising logic and principles of combination.

8.2 Validity and research limitations

Efforts were made to ensure the reliability of the empirical findings and conceptual propositions. As with any research project, this one also comes with

its limitations. The limitations are discussed with reference to the quality criteria of case study research as presented by Yin (2009) and include construct validity, internal validity, external validity and the general reliability of research.

Construct validity concerns with the collection of data, concepts and their operationalisation (Yin 2009). Data collection and the initial identification of the instances that were selected for further analysis were guided by the tentative a priori concepts (Eisenhardt 1989b) that were used as sensitising devices (Klein & Myers 1999). The subsequent process of thematic analysis that unfolded in a highly iterative and interrogative fashion gradually arrived at 15 categories and six themes. Although it is not possible to expose in full detail all intricacies of this process, the resulting themes and categories have been introduced presented with sufficient detail so as to provide a chain of evidence that allows for other researchers to identify comparable clusters of themes and categories. The data collection relied on multiple sources of evidence and frequent triangulation for establishing the validity of proposed themes and categories.

Internal validity concerns with the logical validity of data analysis and the construction of an argument that warrants the conclusions presented by the research (Yin 2009). Several analytic strategies can be employed to this end, such as systematic building of explanation and explanation against predicted patterns (Gibbert et al. 2008). As mentioned above, the unfolding of the process of thematic analysis cannot be explicated here in full detail: it is not feasible to establish exposition in full detail on how the cycles of iterations unfolded. However, the methodology chapter describes the overall unfolding of the process that distilled the large body of empirical evidence from heterogeneous sources into a handful of categories and concepts. Furthermore, the empirical findings and conceptual contributions are presented in way that should allow a reader to retrace the evidence and reconstruct the proposed line of argument. Therefore, the validity of the proposed concepts can be subjected to verification. In addition, the outcomes have been discussed in the light of existing literature to sharpen the boundaries and definitions of the proposed conceptualisations and to demonstrate the ways in which they differ from the current conceptualisations and thereby provide novel and original contribution to the literature on digital innovation.

External validity concerns to what extent the final conclusions of research are expected to be generalisable to the settings that can be considered similar but reside outside the empirical context of this research (Yin 2009; Gibbert et al. 2008). The two conceptual propositions differ in terms of their generalisability. The notion of chains of transformations can be considered as highly generalisable across a range of systems that carry out transformations in a distributed manner. Broadly speaking, it marks the shift form information systems to transformation systems while highlighting the differing underlying logics of transferability, generalisability and verifiability of computational process that serve different purposes. In turn, the generative-integrative mode of development is more constrained and confined to the settings that unfold under particular assumptions concerning the coordination, control and access to knowledge at the level of overall system architectures and their constitutive components.

Finally, the overall reliability of research refers to the absence of errors and minimisation of biases (Gibbert et al. 2008; Yin 2009). In principle, other researchers should be able to reproduce the findings if they would replicate the research project. Considering the efforts that have been made to describe the purpose, process and findings of this research with explicit references to the documentary evidence that is publicly available, in principle, it should be possible for other researchers to replicate the research and reach similar empirical findings and conceptual conclusions.

8.3 Future research

As this research sheds light on the organising logic of complex digital innovation in the context of robot and autonomous systems, it also brings forward a range of topics and phenomena that would warrant further examination to develop a better understanding of the dynamics of innovation revolving around the chains of transformation in their different forms and functions.

Considering that the proposed concepts were developed based on the empirical evidence that represent a single open-source community, the applicability of conceptualisations could be further examined in other empirical contexts. Evaluating the proposed concepts in other settings could validate, refute or

extend these conceptualisations. In the context of robots and autonomous systems, for example, Orocos (Bruyninckx 2001) could be expected to be similar to the distributed architecture of ROS in certain aspects, whereas Jibo,[88] an interactive desktop robot with its encapsulated hardware, application programming interfaces and system development toolkit, presumably resembles more a hardware-device platform that follows the logic of layered modular architecture in the same way as mobile devices do.

Moreover, as this research focuses on the overall characteristics of the organising logic of complex digital innovation at the level of ROS and the related community, a more nuanced view could be obtained by examining some particular instances of complex digital innovation from a close distance. For example, an in-depth case study into a systems development project could help understand and theorise how developers and organisations experience and approach in their day to day practices and processes the challenges that revolve around the specificity of requirements and the distributedness of knowledge and control.

Furthermore, while the complexity of technological systems leads to several technical and engineering challenges, developing understanding of the variety of environmental contingences plays also a significant role. In the end, designing and developing a machine that responds and absorbs environmental contingencies require a deep understanding of the underlying characteristics and dynamics of tasks and task environments. As robots and autonomous systems are deployed into more open-ended and dynamic environments, the methods and approaches for modelling environments in terms of their dynamics and interactional affordances are expected to grow in importance. The extensive use of visualisation and simulation tools demonstrates the need for this, yet more research is needed on establish which ways the knowledge and codification of tasks and task environments intersperses the technological knowledge that dominates the discourse of robots and autonomous systems. To exemplify, whereas an implementation of a management information system often involves the rearrangement of administrative practices to make them compatible with the workflows that are supported by the information system,

[88] www.jibo.com

many physical large-scale environments such the system of road transit cannot be changed but must be learned and adapted into. In this light, the data and models that document the variety of environmental and operational contexts that condition and accept or reject the behaviour of robots and autonomous systems is presumably highly valuable for the developers of robot systems.

Overall, the further research into the computer-controlled behaviour and the chains of transformation in their different forms and functions is necessary to develop better understanding of the machines to which the powers of sense and decision making are increasingly delegated and attributed; it is of paramount importance to develop the methods and tools that provide us with a holistic understanding of the structural and functional aspects of the modern computing technologies. Examining and categorising different kinds of chains of transformation and associated control points could provide better grounding for explaining the social and organisation phenomena that revolves around complex digital innovation. To this end, Floridi's (2008, 2013) method of the levels of abstraction could provide an appropriate high-level framing to begin developing a more detailed framework to captures the dynamics of abstractions, transformations at and across different levels. This would help replace the terms such as artificial intelligence and algorithmic powers with more grounded and operationalisable terms, thereby allow us to pose more tractable questions such as what is being transformed, on what grounds and who controls transformations.

9 References

Aaltonen, A. & Lanzara, G.F., 2015. Building Governance Capability in Online Social Production: Insights from Wikipedia. *Organization Studies*, 36(12), pp.1649–1673.

Abernathy, W.J. & Utterback, J.M., 1978. Patterns of industrial innovation. *Technology Review*, 80(7), pp.2–9.

Ackoff, R.L., 1971. Towards a System of Systems Concepts. *Management Science*, 17(11), pp.661–671.

Ahmed, P.K. & Shepherd, C., 2010. *Innovation Management*, Prentice Hall.

Alaimo, C. & Kallinikos, J., 2017. Computing the everyday: Social media as data platforms. *The Information Society*, 33(4), pp.175–191.

Alexander, C., 1964. *Notes on the Synthesis of Form*, Harvard University Press.

Antonelli, G., 2015. Robotic Research: Are We Applying the Scientific Method? *Frontiers in Robotics and AI*, 2(13), p.28.

Arthur, W.B., 2009. *The Nature of Technology*, Simon and Schuster.

Arthur, W.B., 2007. The structure of invention. *Research Policy*, 36(2), pp.274–287.

Ashby, W.R., 1958. Requisite variety and its implications for the control of complex systems. *Cybernetica*, 1(2), pp.83–89.

Avgerou, C., 2000. Information systems: what sort of science is it? *Omega*, 28(5), pp.567–579.

Bailey, D.E., Leonardi, P.M. & Barley, S.R., 2012. The Lure of the Virtual. *Organization Science*, 23(5).

Baldwin, C.Y. & Clark, K.B., 2000. *Design Rules: The power of modularity*, London: MIT Press.

Baldwin, C.Y. & Woodard, C.J., 2008. The architecture of platforms: A unified view. *Harvard Business School Finance Working Paper*.

Barrett, M. et al., 2012. Reconfiguring Boundary Relations: Robotic Innovations in Pharmacy Work. *Organization Science*, 23(5), pp.1448–1466.

Barrett, M. et al., 2015. Service innovation in the digital age: key contributions and future directions. *MIS Quarterly*, 39(1), pp.135–154.

Bartholet, R.G., Brogan, D.C. & Reynolds, P.F., Jr, 2004. In search of the philosopher's stone: Simulation composability versus component-based software design. In Proceedings of the Fall Simulation Interoperability Workshop.

Bauer, M. & Gaskell, G., 2000. Corpus Construction: a Principle for Qualitative Data Collection. In *Qualitative Researching with Text, Image and Sound.* London: SAGE Publications Ltd, pp. 20–37. Available at: http://methods.sagepub.com/book/qualitative-researching-with-text-image-and-sound.

Bauer, M.W., Gaskell, G. & Allum, N.C., 2000. Quality, Quantity and Knowledge Interests: Avoiding Confusions. In *Qualitative Researching with Text, Image and Sound*. London: SAGE Publications Ltd, pp. 3–17.

Bedau, M.A. & Humphreys, P., 2008. Introduction to Philosophical Perspectives on Emergence. In *Emergence*. Contemporary Readings in Philosophy and Science. The MIT Press, pp. 9–17.

Bekey, G.A., 2005. *Autonomous Robots*, London: MIT Press.

Bertalanffy, von, L., 1950. An outline of general system theory. *The British Journal for the Philosophy of Science*, 1(2), pp.134–165.

Bertalanffy, von, L., 1968. *General System Theory*, New York: George Braziller.

Bharadwaj, A. et al., 2013. Digital Business Strategy: Toward a Next Generation of Insights. *MIS Quarterly*, 37(2), pp.471–482.

Bissell, C., 2009. A History of Automatic Control. In S. Y. Nof, ed. *Springer Handbook of Automation*. Heidelberg: Springer, pp. 53–69.

Bonsignorio, F. & del Pobil, A.P., 2015. Toward Replicable and Measurable Robotics Research [From the Guest Editors]. *IEEE Robotics & Automation Magazine*, 22(3), pp.32–35.

Boulding, K.E., 1956. General Systems Theory-The Skeleton of Science. *Management Science*, 2(3), pp.197–208.

Bowen, G.A., 2009. Document Analysis as a Qualitative Research Method. *Qualitative Research Journal*, 9(2), pp.27–40.

Boyatzis, R.E., 1998. *Transforming qualitative information*, Sage Publications, Inc.

Braun, V. & Clarke, V., 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2), pp.77–101.

Broy, M., Cengarle, M.V. & Geisberger, E., 2012. Cyber-Physical Systems: Imminent Challenges. In Lecture Notes in Computer Science. Springer, Berlin, Heidelberg, pp. 1–28.

Brusoni, S. & Prencipe, A., 2006. Making Design Rules: A Multidomain Perspective. *Organization Science*, 17(2), pp.179–189.

Brusoni, S., Prencipe, A. & Pavitt, K., 2001. Knowledge specialization, organizational coupling, and the boundaries of the firm: why do firms know more than they make? *Administrative Science Quarterly*, 46(4), pp.597–621.

Bruyninckx, H., 2001. Open robot control software: the OROCOS project. In IEEE International Conference on Robotics and Automation. pp. 2523–2528.

Bryman, A., 2015. *Social Research Methods*, Oxford University Press.

Bryson, J., 2010. Cross-paradigm analysis of autonomous agent architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, 12(2), pp.165–189.

Buchanan, R., 1992. Wicked Problems in Design Thinking. *Design Issues*, 8(2), pp.5–21.

Camillus, J.C., 2008. Strategy as a wicked problem. *Harvard Business Review*, 86(5), pp.99–106.

Campagnolo, D. & Camuffo, A., 2009. The concept of modularity in management studies: A literature review. *International Journal of Management Reviews*, 12(3), pp.259–283.

Checkland, P., 2000. Soft systems methodology: a thirty year retrospective. *Systems Research and Behavioral Science*, 16, pp.11–58.

Ciborra, C.U., 1996. The Platform Organization: Recombining Strategies, Structures, and Surprises. *Organization Science*, 7(2), pp.103–118.

Clark, K.B., 1985. The interaction of design hierarchies and market concepts in technological evolution. *Research Policy*, 14(5), pp.235–251.

Conti, M. et al., 2012. Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence. *Pervasive and Mobile Computing*, 8(1), pp.2–21.

Copeland, B.J., 2014. *Turing: Pioneer of the Information Age*, Oxford University Press.

Corbin, J. & Strauss, A., 2008. *Basics of Qualitative Research*, London: SAGE Publications, Inc.

Cousins, S., 2010. ROS on the PR2. *IEEE Robotics & Automation Magazine*, 17(3), pp.23–25.

Cousins, S., 2014. Willow Garage Retrospective [ROS Topics]. *IEEE Robotics & Automation Magazine*, 21(1), pp.16–20.

Crossan, M.M. & Apaydin, M., 2009. A Multi-Dimensional Framework of Organizational Innovation: A Systematic Review of the Literature. *Journal of Management Studies*, 47(6), pp.1154–1191.

Danese, P. & Filippini, R., 2010. Modularity and the impact on new product development time performance. *International Journal of Operations & Production Management*, 30(11), pp.1191–1209.

Davenport, T.H., 1993. *Process Innovation*, Boston: Harvard Business Shool Press.

de Reuver, M., Sørensen, C. & Basole, R.C., 2017. The digital platform: a research agenda. *Journal of Information Technology*, pp.1–12.

Eaton, B. et al., 2015. Distributed Tuning of Boundary Resources: The Case of Apple's iOS Service System. *MIS Quarterly*, 39(1), pp.217–243.

Eisenhardt, K.M., 1989a. Agency Theory: An Assessment and Review. *The Academy of Management Review*, 14(1), p.57.

Eisenhardt, K.M., 1989b. Building Theories from Case Study Research. *The Academy of Management Review*, 14(4), pp.532–550.

Eisenhardt, K.M. & Graebner, M.E., 2007. Theory Building From Cases: Opportunities And Challenges. *Academy of Management Journal*, 50(1), pp.25–32.

Eisenhardt, K.M., Graebner, M.E. & Sonenshein, S., 2016. Grand Challenges and Inductive Methods: Rigor without Rigor Mortis. *Academy of Management Journal*, 59(4), pp.1113–1123.

Ernst, D., 2005. Limits to Modularity: Reflections on Recent Developments in Chip Design. *Industry & Innovation*, 12(3), pp.303–335.

Ethiraj, S.K. & Levinthal, D., 2004. Modularity and Innovation in Complex Systems. *Management Science*, 50(2), pp.159–173.

Fichman, R.G., Santos, Dos, B.L. & Zheng, Z., 2014. Digital innovation as a fundamental and powerful concept in the information systems curriculum. *MIS Quarterly*, 38(2), pp.329–353.

Floridi, L., 2008. The Method of Levels of Abstraction. *Minds and Machines*, 18(3), pp.303–329.

Floridi, L., 2013. *The Philosophy of Information*, OUP Oxford.

Flyvbjerg, B., 2013. Case Study. In N. K. Denzin & Y. S. Lincoln, eds. *Strategies of qualitative inquiry*. Thousand Oaks, CA: SAGE, pp. 169–205.

Foote, T., 2013. Tf: The transform library. In 2013 IEEE Conference on Technologies for Practical Robot Applications, TePRA 2013. Woburn, MA.

Garcia, R. & Calantone, R., 2002. A Critical Look at Technological Innovation Typology and Innovativeness Terminology. *The Journal of Product Innovation Management*, (19), pp.110–132.

Garud, R. & Kumaraswamy, A., 1995. Technological and organisational designs for realizing economies of substitution. *Strategic Management Journal*, 16, pp.93–109.

Garud, R., Jain, S. & Tuertscher, P., 2008. Incomplete by Design and Designing for Incompleteness. *Organization Studies*, 29(3), pp.351–371.

Gates, B., 2007. A Robot in Every Home. *Scientific American*, 296(1), pp.58–65.

Gerkey, B. & Conley, K., 2011. Robot Developer Kits [ROS Topics]. *IEEE Robotics & Automation Magazine*, 18(3), pp.16–16.

Ghazawneh, A. & Henfridsson, O., 2012. Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal*, 23(2), pp.173–192.

Gibbert, M., Ruigrok, W. & Wicki, B., 2008. What passes as a rigorous case study? *Strategic Management Journal*, 29(13), pp.1465–1474.

Glanville, R., 1997. A Ship without a Rudder. *Problems of Excavating Cybernetics and Systems*, pp.1–10.

Glaser, B.G. & Strauss, A.L., 1967. *The Discovery of Grounded Theory*, Transaction Publishers.

Goldin, D., Smolka, S.A. & Wegner, P., 2006. *Interactive Computation*, Springer.

Gregor, S., 2006. The Nature of Theory in Information Systems. *MIS Quarterly*, 30(3), pp.611–642.

Gregory, D., 2011. From a View to a Kill. *Theory, Culture & Society*, 28(7-8), pp.188–215.

Grier, D.A., 2005. *When Computers Were Human*, Woodstock: Princeton University Press.

Grover, V. & Lyytinen, K., 2015. New State of Play in Information Systems Research: The Push to the Edges. *MIS Quarterly*, 39(2), pp.271–296.

Guizzo, E., 2014. Microsoft Shuts Down Its Robotics Group. *IEEE Spectrum*. Available at: http://spectrum.ieee.org/automaton/robotics/robotics-software/microsoft-shuts-down-its-robotics-group [Accessed April 11, 2016].

Hanseth, O. & Lyytinen, K., 2010. Design theory for dynamic complexity in information infrastructures: the case of building internet. *Journal of Information Technology*, 25(1), pp.1–19.

Hayles, N.K., 1999. *How We Became Posthuman: Virtual Bodies in Cybernetics, Literature, and Informatics*, Chicago: University of Chicago Press.

Henderson, R.M. & Clark, K.B., 1990. Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, 35(1), pp.9–30.

Henfridsson, O., Mathiassen, L. & Svahn, F., 2014. Managing technological change in the digital age: the role of architectural frames. *Journal of Information Technology*, 29(1), pp.27–43.

Hernandez, C. et al., 2017. Team Delft's Robot Winner of the Amazon Picking Challenge 2016. In S. Behnke et al., eds. Lecture Notes in Computer Science.

RoboCup 2016: Robot World Cup XX. Cham: Springer International Publishing, pp. 613–624.

Hintikka, J., 1999. The Role of Logic in Argumentation. In *Inquiry as Inquiry: A Logic of Scientific Discovery*. Dordrecht: Springer Netherlands, pp. 25–46.

Hippel, Von, E., 1990. Task partitioning: An innovation process variable. *Research Policy*, 19(5), pp.407–418.

Hobday, M., 1998. Product complexity, innovation and industrial organisation. *Research Policy*, 26(6), pp.689–710.

Honderich, T., 2005. *The Oxford Companion to Philosophy*, Oxford: Oxford University Press.

Hylving, L. & Schultze, U., 2013. Evolving The Modular Layered Architecture In Digital Innovation: The Case Of The Car's Instrument Cluster. *Thirty Fourth International Conference on Information Systems*, pp.1–17.

Iman, N., 2016. Modularity matters: a critical review and synthesis of service modularity. *International Journal of Quality and Service Sciences*, 8(1), pp.38–52.

Iñigo-Blasco, P. et al., 2012. Robotics software frameworks for multi-agent robotic systems development. *Robotics and Autonomous Systems*, 60(6), pp.803–821.

Kallinikos, J., 2012. Form, Function, and Matter: Crossing the Border of Materiality. In P. M. Leonardi, B. A. Nardi, & J. Kallinikos, eds. *Materiality and Organizing*. Social Interaction in a Technological World. Oxford University Press, pp. 67–87.

Kallinikos, J., 2002. Reopening the black box of technology artifacts and human agency. *Twenty-Third International Conference on Information Systems*, pp.287–294.

Kallinikos, J., 2005. The order of technology: Complexity and control in a connected world. *Information and Organization*, 15(3), pp.185–202.

Kallinikos, J., Aaltonen, A. & Marton, A., 2013. The ambivalent ontology of digital artifacts. *MIS Quarterly*, 37(2), pp.357–370.

Kallinikos, J., Hasselbladh, H. & Marton, A., 2013. Governing social practice. *Theory and Society*, 42(4), pp.395–421.

Karhu, K., Tang, T. & Hämäläinen, M., 2014. Analyzing competitive and collaborative differences among mobile ecosystems using abstracted strategy networks. *Telematics and Informatics*, 31(2), pp.319–333.

Kauffman, S.A., 1970. Articulation of Parts Explanation in Biology and the Rational Search for Them. *PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association*, 1970, pp.257–272.

Klaus, H., Rosemann, M. & Gable, G.G., 2000. What is ERP? *Information Systems Frontiers*, 2(2), pp.141–162.

Klein, H.K. & Myers, M.D., 1999. A set of principles for conducting and evaluating interpretive field studies in information systems. *MIS Quarterly*, 23(1), pp.67–93.

Kohlbrecher, S. et al., 2013. Overview of team ViGIR's approach to the Virtual Robotics Challenge. In 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) IS. pp. 1–2.

Kohlbrecher, S. et al., 2015. Human-robot Teaming for Rescue Missions: Team ViGIR's Approach to the 2013 DARPA Robotics Challenge Trials *Journal of Field Robotics*, 32(3), pp.352–377.

Kramer, J. & Scheutz, M., 2006. Development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2), pp.101–132.

Langlois, R.N. & Robertson, P.L., 1992. Networks and innovation in a modular system: Lessons from the microcomputer and stereo component industries. *Research Policy*, 21(4), pp.297–313.

Lee, A.S., 2010. Retrospect and prospect: information systems research in the last and next 25 years. *Journal of Information Technology*, 25(4), pp.336–348.

Lee, A.S. & Baskerville, R.L., 2003. Generalizing generalizability in information systems research. *Information Systems Research*, 14(3), pp.221–243.

Lee, J. & Berente, N., 2012. Digital Innovation and the Division of Innovative Labor: Digital Controls in the Automotive Industry. *Organization Science*, 23(5), pp.1428–1447.

Lyyra, A.K. & Koskinen, K.M., 2016. The Ambivalent Characteristics of Connected, Digitised Products: Case Tesla Model S. In C. Keller et al., eds. *Nordic Contributions in IS Research*. Lecture Notes in Business Information Processing. Springer International Publishing, pp. 57–69.

Lyytinen, K., Yoo, Y. & Boland, R.J., Jr., 2015. Digital product innovation within four classes of innovation networks. *Information Systems Journal*, 26(1), pp.47–75.

Mack, C.A., 2011. Fifty Years of Moore's Law. *IEEE Transactions on Semiconductor Manufacturing*, 24(2), pp.202–207.

Maes, P., Guttman, R.H. & Moukas, A.G., 1999. Agents that buy and sell. *Communications of the ACM*, 42(3), pp.81–91.

Manyika, J. et al., 2013. *Disruptive technologies: Advances that will transform life, business, and the global economy*, McKinsey Global Institute.

Mason, M., 2012. Creation Myths: The Beginnings of Robotics Research. *IEEE Robotics & Automation Magazine*, 19(2), pp.72–77.

Matook, S. & Brown, S.A., 2016. Characteristics of IT artifacts: a systems thinking-based framework for delineating and theorizing IT artifacts. *Information Systems Journal*, 17(3), pp.661–38.

Maturana, H.R. & Varela, F.J., 1992. *The tree of knowledge*, Boston, MA: Shambhala Publications, Inc.

Meadows, D.H., 2009. *Thinking in Systems*, Earthscan. Available at: http://ir.nmu.org.ua/bitstream/handle/123456789/129200/2ee4a14a158e824b867e07ad95005643.pdf?sequence=1.

Merali, Y. & Allen, P., 2011. Complexity and Systems Thinking. In *The SAGE Handbook of Complexity and Management*. SAGE, pp. 31–52.

Metta, G., Fitzpatrick, P. & Natale, L., 2006. YARP: Yet Another Robot Platform. *International Journal of Advanced Robotic Systems*, 3(1), pp.43–48.

Miller, R., Hobday, M. & Leroux-Demers, T., 1995. Innovation in complex systems industries: the case of flight simulation. *Industrial and corporate change*, 4(2), pp.363–400.

Mindell, D.A., 2015. *Our Robots, Ourselves*, New York: Penguin.

Mitchell, M., 2009. *Complexity: A Guided Tour*, Oxford: Oxford University Press.

Murmann, J.P. & Frenken, K., 2006. Toward a systematic framework for research on dominant designs, technological innovations, and industrial change. *Research Policy*, 35(7), pp.925–952.

Nambisan, S. et al., 2017. Digital innovation management: Reinventing Management Research in a Digital World. *MIS Quarterly*, 41(1), pp.223–238.

Ng, A.Y. & Khatib, O., 2006. *CRI: the Stanford AI STAIR Robot Project*,

Ng, A.Y. et al., 2008. STAIR: The STanford Artificial Intelligence Robot project. In Snowbird. pp. 1–2.

Nolte, D.D., 2015. *Introduction to Modern Dynamics*, Oxford: Oxford University Press.

O'Kane, J.M., 2014. A gentle introduction to ROS.

Olsen, S., 2006. Microsoft unveils public robotics software. *cnet.com*. Available at: https://www.cnet.com/uk/news/microsoft-unveils-public-robotics-software/ [Accessed April 11, 2016].

Parnas, D.L., 1972. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12), pp.1053–1058.

Perrow, C., 1984. *Normal Accidents*, New York: Basic Books.

Pot, E. et al., 2009. Choregraphe: a graphical tool for humanoid robot programming. In RO-MAN 2009 - The 18th IEEE International Symposium on Robot and Human Interactive Communication IS . IEEE, pp. 46–51.

Prencipe, A., 2000. Breadth and depth of technological capabilities in CoPS: the case of the aircraft engine control system. *Research Policy*, 29(7-8), pp.895–911.

Quigley, M. et al., 2009. ROS: an open-source Robot Operating System. In ICRA workshop on open source software 3(3.2), pp. 1-6.

Quigley, M., Berger, E. & Ng, A.Y., 2007. STAIR: Hardware and Software Architecture. *AAAI Robotics Workshop*, pp.1–7.

Rittel, H. & Webber, M.M., 1973. Dilemmas in a general theory of planning. *Policy sciences*, 4(2), pp.155–169.

Rossano, G.F. et al., 2013. Easy robot programming concepts: An industrial perspective. In Automation Science and Engineering (CASE), 2013 IEEE International Conference on. IEEE, pp. 1119–1126.

Russell, S.J. & Norvig, P., 2010. *Artificial Intelligence*, New Jersey: Prentice Hall.

Ruttan, V.W., 1959. *Usher and Schumpeter on invention, innovation, and technological change*, The quarterly journal of economics.

Salvador, F., 2007. Toward a Product System Modularity Construct: Literature Review and Reconceptualization. *IEEE Transactions on Engineering Management*, 54(2), pp.219–240.

Sambamurthy, V. & Zmud, R.W., 2000. Research Commentary: The organising Logic for an Enterprise's IT activities in the Digital Era – A prognosis of Practice and a call for Research. *Information Systems Research*, 11(2), pp.105–114.

Sanchez, R. & Mahoney, J.T., 1996. Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management Journal*, 17(S2), pp.63–76.

Sanfeliu, A., Hagita, N. & Saffiotti, A., 2008. Network robot systems. *Robotics and Autonomous Systems*, 56(10), pp.793–797.

Schilling, M.A., 2000. Toward a General Modular Systems Theory and Its Application to Interfirm Product Modularity. *The Academy of Management Review*, 25(2), pp.312–334.

Schilling, M.A. & Steensma, H.K., 2001. The Use of Modular Organizational Forms: an Industry-Level Analysis. *Academy of Management Journal*, 44(6), pp.1149–1168.

Schumpeter, J.A., 1983. *The Theory of Economic Development*, Transaction Publishers.

Scott, J., 2014. *A Matter of Record: Documentary Sources in Social Research*, John Wiley & Sons.

Seddon, P.B. & Scheepers, R., 2015. Generalization in IS research: a critique of the conflicting positions of Lee & Baskerville and Tsang & Williams. *Journal of Information Technology*, 30(1), pp.30–43.

Seddon, P.B. & Scheepers, R., 2012. Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples. *European Journal of Information Systems*, 21(1), pp.6–21.

Shaikh, M. & Henfridsson, O., 2017. Governing open source software through coordination processes. *Information and Organization*, 27(2), pp.116–135.

Shneiderman, B. & Maes, P., 1997. Direct manipulation vs. interface agents. *interactions*, 4(6), pp.42–61.

Siciliano, B. & Khatib, O., 2008. *Springer Handbook of Robotics*, Stanford, CA: Springer.

Silverman, D., 2015. *Interpreting Qualitative Data*, London: SAGE Publications.

Simon, H.A., 1962. The Architecture of Complexity. *Proceedings of the American Philosophical Society*, 106(6), pp.467–482.

Simon, H.A., 1996. *The Sciences of the Artificial*, MIT Press.

Sosa, M.E., Eppinger, S.D. & Rowles, C.M., 2004. The Misalignment of Product Architecture and Organizational Structure in Complex Product Development. *Management Science*, 50(12), pp.1674–1689.

Spiegel, M., Reynolds, P.F., Jr & Brogan, D.C., 2005. A Case Study of Model Context for Simulation Composability and Reusability. *Winter Simulation Conference*, pp.437–444.

Sterling, B., 2013. Open-source Robot Operating System. *wired.com*. Available at: https://www.wired.com/2013/10/open-source-robot-operating-system/ [Accessed July 10, 2017].

Suarez, F.F., 2004. Battles for technological dominance: an integrative framework. *Research Policy*, 33(2), pp.271–286.

Svahn, F., Mathiassen, L. & Lindgren, R., 2017. Embracing Digital Innovation in Incumbent Firms: How Volvo Cars Managed Competing Concerns. *MIS Quarterly*, 41(1), pp.239–253.

Swade, D., 2002. *The Difference Engine: Charles Babbage and the Quest to Build the First Computer*, Penguin Group USA.

Tanenbaum, A.S. & Bos, H., 2014. *Modern Operating Systems*, Essex: Pearson.

Thrun, S. et al., 2006. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9), pp.661–692.

Tilson, D., Lyytinen, K. & Sørensen, C., 2010. Research Commentary—Digital Infrastructures: The Missing IS Research Agenda. *Information Systems Research*, 21(4), pp.748–759.

Tolk, A., Diallo, S.Y. & Turnitsa, C.D., 2007. Applying the levels of conceptual interoperability model in support of integratability, interoperability, and composability for system-of-systems engineering. *Journal of Systemics, Cybernetics and Informatics*, 5(5), pp.65–74.

Ulrich, K.T., 1995. The role of product architecture in the manufacturing firm. *Research Policy*, 24(3), pp.419–440.

Ulrich, K.T. & Eppinger, S.D., 2012. *Product Design and Development*, New York: McGraw-Hill.

Urquhart, C., 2013. *Grounded Theory for Qualitative Research*, London: SAGE.

Utterback, J.M. & Abernathy, W.J., 1975. A dynamic model of process and product innovation. *Omega*, 3(6), pp.639–656.

Vargo, S.L. & Lusch, R.F., 2004. Evolving to a New Dominant Logic for Marketing. *Journal of Marketing*, 68(1), pp.1–17.

Varian, H.R., 2010. Computer Mediated Transactions. *The American Economic Review*, 100(2), pp.1–10.

Walsham, G., 1995. Interpretive case studies in IS research: nature and method. *European Journal of Information Systems*, 4(2), pp.74–81.

Wegner, P., 1997. Why interaction is more powerful than algorithms. *Communications of the ACM*, 40(5), pp.80–91.

Weick, K.E., 1989. Theory Construction as Disciplined Imagination. *The Academy of Management Review*, 14(4), pp.516–531.

Wiener, N., 1965. *Cybernetics, or control and communication in the animal and the machine (2nd ed.)*, Cambridge: MIT Press.

Wilson, D., 1969. Forms of hierarchy: a selected bibliography. *General Systems*, 14, pp.3–15.

Wimsatt, W.C., 1972. Complexity and Organization. In K. F. Schaffner & R. S. Cohen, eds. PSA Proceedings of the Biennial Meeting of the Philosophy of Science Association. Dordrecht, pp. 67–86. Available at: http://www.jstor.org/stable/3698961.

Wimsatt, W.C., 1994. The ontology of complex systems: levels of organization, perspectives, and causal thickets. *Canadian Journal of Philosophy*, Supplementary Volume, Jan 1, 1994, pp.207–274.

Wolf, W., 2009. Cyber-physical Systems. *Computer*, 42(3), pp.88–89.

Wooldridge, M. & Jennings, N.R., 1995. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(2), pp.115–152.

Wray, K.H., Pineda, L. & Zilberstein, S., 2016. Hierarchical Approach to Transfer of Control in Semi-Autonomous Systems: (Extended Abstract). In Proceedings of the 2016 International Conference on Autonomous Agents \&\#38; Multiagent Systems. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, pp. 1285–1286.

Wyrobek, K.A. et al., 2008. Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot. In IEEE International Conference on Robotics and Automation. IEEE, pp. 2165–2170.

Yin, R.K., 2009. *Case study research: design and methods; 4th ed*, London: Sage.

Yoo, Y., 2012a. Digital Materiality and the Emergence of an Evolutionary Science of the Artificial. In *Materiality and Organizing*. Social Interaction in a Technological World. Oxford University Press, pp. 134–154.

Yoo, Y., 2012b. The Tables Have Turned: How Can the Information Systems Field Contribute to Technology and Innovation Management Research? *Journal of the Association for Information Systems*, 14(5), pp.227–236.

Yoo, Y., Henfridsson, O. & Lyytinen, K., 2010. Research Commentary--The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. *Information Systems Research*, 21(4), pp.724–735.

Yoo, Y., Lyytinen, K. & Majchrzak, A., 2012. Organizing for Innovation in the Digitized World. *Organization Science*, 23(5), pp.1398–1408.

Yoo, Y., Lyytinen, K., et al., 2010. Unbounded innovation with digitalization: A case of digital camera. In Annual Meeting of the Academy of Management. pp. 1–41.

Zimbardo, P.G., Maslach, C. & Haney, C., 2000. Reflections on the Stanford prison experiment: Genesis, transformations, consequences. In T. Blass, ed. *Obedience to Authority Current Perspectives on the Milgram Paradigm*. Mahwah, N.J.: Erlbaum Mahwah, NJ, pp. 193–237.

Zittrain, J., 2008. *The Future of the Internet--And How to Stop It*, Yale University Press.

Zittrain, J.L., 2006. The Generative Internet. *Harvard Law Review*, 119(7), pp.1974–2040.

10 Appendix A

Links to the sources of documentary evidence.

Source	Link
Stanford Artificial Intelligence Robot Project	http://stair.stanford.edu
Personal Robotics Programme at Stanford	http://personalrobotics.stanford.edu
Willow Garage website	http://www.willowgarage.com
ROS website	http://www.ros.org
ROS wiki	http://wiki.ros.org
ROS answers	http://answers.ros.org
ROS discussion forum	http://discourse.ros.org
ROS users mailing lists	http://lists.ros.org/pipermail/ros-users/
ROS release mailing lists	http://lists.ros.org/pipermail/ros-release/
ROSCon 2012	http://roscon.ros.org/2012/
ROSCon 2013	http://roscon.ros.org/2013/
ROSCon 2014	http://roscon.ros.org/2014/
ROSCon 2015	http://roscon.ros.org/2015/
ROScon 2016	http://roscon.ros.org/2016/
ROS 2 design website	http://design.ros2.org
ROS 2 design discussion	http://groups.google.com/forum/?fromgroups#!forum/ros-sig-ng-ros
ROS-Industrial	http://rosindustrial.org
Open Source Robotics Foundation	https://www.osrfoundation.org
STAIR boldly steps into the future of robotics	https://engineering.stanford.edu/news/stair-boldly-steps-future-robotics
Interview: Scott Hassan on Willow Garage and the Future of Suitable Technologies	https://spectrum.ieee.org/automaton/robotics/home-robots/interview-scott-hassan-on-willow-garage-and-the-future-of-suitable-tech
Willow Garage's Last Days	https://www.bloomberg.com/news/articles/2014-02-20/robotics-research-lab-willow-garage-shuts-down
Willow Garage changing	http://robohub.org/willow-garage-changing/

This woman makes robots. And no one is going to stop her	https://www.wired.com/2015/05/this-woman-makes-robots-and-no-one-is-going-to-stop-her/#.4axl7ttds
How a billionaire who wrote Google's original code created a robot revolution	http://uk.businessinsider.com/a-look-back-at-willow-garage-2016-2
Willow Garage Retrospective	http://ieeexplore.ieee.org/abstract/document/6763186/

11 Appendix B

The appendices B to G lists the ROSCon conference presentations by theme. Links to the websites that lead to the links of individual presentations can be found in Appendix A.

Theme: Communication system					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
CO1	2012	Keynote: ROS: Past, present, and future	65 mins.	ROS	Messaging; Coordination; Connectors
CO2	2012	Introduction to rosjava	38	rosjava	Connectors; Infrastructure and tools
CO3	2012	ROS on Windows	19	catkin, Windows	Connectors; Infrastructure and tools
CO4	2012	The current state and future of multi-master, multi-robot systems using ROS	34	roscore(s)	Coordination
CO5	2012	Writing Hardware Drivers	40	Hardware drivers	Connectors
CO6	2012	Robot Web Applications	22	rosbridge	Connectors; Infrastructure and tools
CO7	2012	Using Open Sound Control Hardware and Software with ROS	13	touchOSC, iOS	Connectors
CO8	2013	ROS Extrospection – Multimaster and Beyond	14	roscore(s)	Coordination
CO9	2013	Reliable Robotics – ROS Diagnostics++	16	ros_comm	Messaging; Coordination
CO10	2013	Networking for ROS Users	12	ros_comm	Messaging; Coordination
CO11	2013	Android sensors driver	12	Android	Connectors
CO12	2013	Creating web-enabled robots with Robot Web Tools	20	rosbridge2	Connectors

Theme: Communication system					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
CO13	2013	The next (big) step for the ROS middleware	19 mins.	ROS2	Messaging; Coordination
CO14	2013	uROSnode – running ROS on microcontrollers	17	uROSnode	Connectors
CO15	2013	Bridging ROS to Embedded Systems: A Survey	16	connections to hardware	Connectors
CO16	2013	Introducing rosc	16	rosc	Connectors
CO17	2014	Next-generation ROS: Building on DDS	23	ROS2	Messaging; Coordination
CO18	2014	Serious rosserial	14	rosserial	Connectors
CO19	2014	ROS 2.0: Developer preview	44	ROS2	Messaging; Coordination
CO20	2014	ROS support from MATLAB	28	Matlab	Connectors; Infrastructure and tools
CO21	2015	ROS 2 on “small” embedded systems	23	ROS2	Messaging; Coordination; Connectors
CO22	2015	State of ROS 2 - demos and the technology behind	50	ROS2	Messaging; Supporting
CO23	2015	Real-time Performance in ROS 2	40	ROS2	Messaging; Coordination
CO24	2015	ROS android_ndk: What? Why? How?	10	Android NDK	Connectors; Infrastructure and tools
CO25	2016	ROS 2 Update	45	ROS2	Messaging; Coordination
CO26	2016	Adaptive Fault Tolerance on ROS: A Component-Based Approach	20	roscore, ros_comms	Messaging; Coordination

Theme: Communication system					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
CO27	2016	The Intuitive ROS UI: FlexGui 4.0 – introduction and industrial applications	16 mins.	Flex Gui, a user interface	Connectors; Infrastructure and tools
CO28	2016	{,S}ROS: Securing ROS over the wire, in the graph, and through the kernel	20	{,S}ROS	Messaging; Supporting
CO29	2016	Evaluating the resilience of ROS2 communication layer	20	DDS, ROS2	Messaging; Supporting
CO30	2016	RTROS – A real-time extension to the Robot Operating System	21	RTROS	Messaging; Supporting

12 Appendix C

Theme: ROS community and software development					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
SE1	2012	Opening Remarks	10 mins.	ROSCon	Knowledge transfer
SE2	2012	The ROS wiki how to make the best use of it	18	ROS wiki	Knowledge transfer
SE3	2012	Measuring and Tracking Code Quality in ROS	22	ROS Ecosystem	Knowledge transfer
SE4	2012	Teaching Robotics with ROS: Experiences, Suggestions, and Tales of Woe	20	ROS Ecosystem	Knowledge transfer
SE5	2012	Closing remarks	4	ROSCon	Knowledge transfer
SE6	2013	Opening remarks	17	ROSCon	Knowledge transfer
SE7	2013	ROS-Industrial, An Open Source Case Study	19	ROS-Industrial	Knowledge transfer
SE8	2013	Why Industrial Robot Manufacturers Should Care About ROS	21	Yaskawa robots	Knowledge transfer
SE9	2013	The ROS Ecosystem: How are We Doing?	19	ROS Ecosystem	Knowledge transfer
SE10	2013	Roles and Responsibilities of a Package Maintainer	10	ROS Ecosystem	Infrastructure and tools
SE11	2013	ROS and Rock: mixing Orocos components and ROS nodes into model-driven toolchain	18	Rock-Orocos, a development toolchain	Knowledge transfer; Infrastructure and tools
SE12	2013	Improve your ROS code with Model-Driven-Engineering and save development time while doing it	19	BRIDE, a development toolchain	Knowledge transfer; Infrastructure and tools

Theme: ROS community and software development					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
SE13	2013	Understanding and using Catkin	41 mins.	catkin, a build system	Infrastructure and tools
SE14	2014	Opening remarks	14	ROSCon	Knowledge transfer
SE15	2014	The ROS ecosystem: Impact, insights, and improvements	25	ROS community	Knowledge transfer
SE16	2014	Continuous integration for ROS in commercial environments	20	buildbot-ros, a private build system	Infrastructure and tools
SE17	2014	Closing remarks	5	ROSCon	Knowledge transfer
SE18	2015	Opening Remarks	11	ROSCon	Knowledge transfer
SE19	2015	ROS for education and applied research: practical experiences	16	ROS	Knowledge transfer
SE20	2015	Bringing ROS to the factory floor: a status report on the ROS-Industrial initiative	37	ROS-Industrial	Knowledge transfer
SE21	2015	Commercial models for the robot generation	45	Ubuntu Snappy, dependency management	Infrastructure and tools
SE22	2015	ROS + Docker: Enabling Repeatable, Reproducible, and Deployable robotic software via Linux Containers	20	Docker, dependency management	Infrastructure and tools
SE23	2015	Docker-based ROS Build Farm	19	private build system (Bosch)	Infrastructure and tools
SE24	2015	Closing remarks	5	ROSCon	Knowledge transfer

Theme: ROS community and software development					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
SE25	2016	Opening Remarks	10 mins.	ROSCon	Knowledge transfer
SE26	2016	ROS-Industrial turns four and expands worldwide	20	ROS-Industrial	Knowledge transfer
SE27	2016	ROS-‘X’ – Focused Initiatives	24	ROS-'X', domain-specific initiatives	Knowledge transfer
SE28	2016	The ROS build farm - what it can do for me	41	Public ROS build system by OSRF	Infrastructure and tools
SE29	2016	Robust Deployment with ROS Bundles	25	private build system (Clearpath)	Infrastructure and tools
SE30	2016	Closing remarks	11	ROSCon	Knowledge transfer

13 Appendix D

Theme: Robot systems					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
RS1	2012	Using ROS on Field Robotic Experiments in Remote Locations	20 mins.	Field robotics	Research Robots
RS2	2012	ROS for Humanoid Robots	23	NAO, a humanoid robot	Research Robots
RS3	2012	“Moe” The Autonomous Lawnmower	18	Moe, a lawnmovign robot	Research Robots
RS4	2012	Keynote: Architecting Real-time Control of Robonaut 2 using ROS and Orocos	57	Robonaut2, a humanoid robot	Research Robots
RS5	2013	Project AUTOMATE at MIT Lincoln Laboratories	20	A multi-robot system	Research Robots
RS6	2013	Real world indoor & outdoor navigation experiences with ROS	15	Robotnik, mobile base	Research Robots
RS7	2013	Hi Richard – Personalize your Robot with the cob_people_perception stack	16	Care-O-bot, research platform	Research Robots
RS8	2013	Understanding the RoboEarth Cloud	25	RoboEarth project	Research Robots
RS9	2014	Development of dual arm mobile manipulation systems for small part assembly tasks	45	PRACE project at IPA	Research Robots
RS10	2014	EuRoC – The European Robotic Challenges	17	EuRoC competitions	Research Robots
RS11	2014	How ROS works together with the mining industry in i2Mine project	21	i2mine project	Research Robots

Theme: Robot systems					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
RS12	2014	Practical experiences using ROS to build a three axis pick and place assembly robot	25 mins.	A proto-board maker	Productive applications
RS13	2014	ROS in space	40	Robonaut2, a humanoid robot	Research Robots
RS14	2014	Control and perception architecture for the tele-operation of the humanoid robot COMAN	17	Coman, a humanoid robot	Research Robots
RS15	2015	An Introduction to Team ViGIR's Open Source Software and DRC Post Mortem	47	Team ViGir's DRC robot	Research Robots
RS16	2015	Automated Driving with ROS at BMW	29	Cars, Automated driving	Research Robots
RS17	2015	Maru and Toru: Item-specific logistics solutions based on ROS	19	Magazino, warehouse robots	Productive applications
RS18	2015	Accelerating Your Robotics Startup with ROS	20	Fetch, warehouse robots	Productive applications
RS19	2016	Plan to Win with MoveIt! - Lessons learnt from the Amazon Picking Challenge 2016	20	A pick and place robot for warehouses	Research Robots
RS20	2016	ANYmal at the ARGOS Challenge: Tools and Experiences from the Autonomous Inspection of Oil & Gas Sites with a Legged Robot	36	ANYmal, Oil and Gas production site	Research Robots
RS21	2016	Agricultural Robotics with ROS at Bosch: From the internet of fields to the internet of plants	23	Deepfield Robotics (Bosch),	Productive applications

14 Appendix E

Theme: Transformation systems					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
TR1	2012	URDF and You	45 mins.	URDF	Coordinate transformation
TR2	2012	Motion Planning in ROS	43	MoveIt!	Coordinate transformation; Representational transformation
TR3	2012	Understanding tf	40	tf	Coordinate transformation
TR4	2012	Understanding the Kinect	42	Kinect, OpenCV, OpenNI	Representational transformation
TR5	2013	Keynote: MoveIt!	39	Moveit!	Coordinate transformation; Representational transformation
TR6	2013	3D Mapping with OctoMap	40	OctoMap	Coordinate transformation; Representational transformation
TR7	2013	Object Recognition Kitchen	23	Ecto	Representational transformation
TR8	2013	Taking advantage of tf2 in single and multi-robot cases	13	tf	Coordinate transformation
TR9	2013	Converting SolidWorks Parts and Assemblies to ROS Friendly Files	14	URDF; sw2urdf	Coordinate transformation
TR10	2014	ros_control: An overview	45	ros_control	Representational transformation
TR11	2014	Navigation illumination: Shedding light on the ROS navstack	45	navigation	Representational transformation

Theme: Transformation systems					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
TR12	2014	ROS-Industrial calibration	27 mins.	Calibration	Coordinate transformation
TR13	2015	MoveIt! Strengths, Weaknesses, and Developer Insight	45	MoveIt!	Coordinate transformation; Representational transformation
TR14	2015	Phobos - Robot Model Development on Steroids	15	URDF; Phobos	Coordinate transformation
TR15	2015	The Descartes Planning Library for Semi-Constrained Cartesian Trajectories	21	Descartes	Coordinate transformation
TR16	2015	Working with the robot_localization Package	20	robot_localisation	Coordinate transformation; Representational transformation
TR17	2016	Robot calibration	19	robot_calibration	Coordinate transformation

15 Appendix F

Theme: Visualisation and testing systems					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
VI1	2012	The Gazebo Simulator as a Development Tool in ROS	46 mins.	Gazebo	Simulation
VI2	2013	Introducing the MORSE simulator	22	Morse	Simulation
VI3	2013	From simulation to real robots	9	USARSim	Simulation
VI4	2013	Using ROS with Webots	12	Webots	Simulation; Development infrastructure and tools
VI5	2013	CloudSim: your own ROS robot army in the cloud	20	Cloudsim, Gazebo	Simulation
VI6	2013	RQT Framework and Best Practices	25	Qt, rqt	Visualisation
VI7	2013	Robot Performance Analysis from Automatically Recorded Data	32	MongoDB, ROS	Data management; Visualisation
VI8	2014	Comparison of rigid body dynamic simulators for robotic simulation in Gazebo	26	Gazebo	Simulation
VI9	2014	Cognitive Robotics Architecture for Tightly-coupled Experiments and Simulation (CRATES)	25	QRSIM, a drone platform	Simulation; Transformation
VI10	2015	Mapviz: An Extensible 2D Visualization Tool for Automated Vehicle	15	mapviz	Visualisation
VI11	2015	ROS-driven user applications in idempotent environments	17	Liquid Galaxy	Visualisation
VI12	2016	What's new in Gazebo? Upgrading your simulation user experience!	39	Gazebo	Simulation
VI13	2016	Cloudy with a Chance of Simulation	21	Gazebo. CloudSim	Simulation

Theme: Visualisation and testing systems					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
VI14	2016	Robotics Benchmarking with ROS	20 mins.	ConstructSim	Simulation
VI15	2016	Physical Continuous Integration — CI on Real Robots!	20	Fetch, a warehouse mock-up	Simulation
VI16	2016	Bagbunker - Tool for Large Data Storage, Analysis, Viewing and Testing	23	MARV, rosbag	Data management; Visualisation

16 Appendix G

Theme: Physical embodiments					
Code	**Year**	**Title of presentation**	**Duration**	**Subsystem**	**Categories**
PE1	2013	Turtlebot 2 – the new standard hardware reference platform	14 mins.	TurtleBot, a teaching platform	Platform; Actuators: Sensors
PE2	2013	ROSifying Robots: Tips, Tricks, and Lesson Learned	10	Any robot hardware	Platform; Actuators: Sensors
PE3	2015	ROS on DroneCode Systems	15	PX4, drone platform	Platform; Actuators: Sensors
PE4	2015	Introducing ROS-RealSense: 3D empowered Robotics Innovation Platform	15	Intel RealSense sensors	Sensors
PE5	2016	Introducing the Turtlebot3	20	TurtleBot a teaching platform	Platform; Actuators: Sensors
PE6	2016	Introducing Intel RealSense Robotics All-in-one Perception Device	22	Intel RealSense sensors	Sensors
PE7	2016	Introducing H-ROS, the Hardware Robot Operating System	22	H-ROS, ROS hardware components	Sensors; Actuators
PE8	2016	A robust flying platform for ROS developers	22	DJI, drone platform	Platform; Actuators: Sensors

17 Appendix H

Events and non-participant observation documented in field notes.

Code	Year	Description	Event and location
EO1	2015	Workshop: Research and innovation camps	European Robotics Forum 2015 in Vienna, Austria
EO2	2015	Workshop: Hardware and software modularity and interoperability in Service Robotics: Towards standardisation	
EO3	2015	Workshop: Towards new Robotics Software	
EO4	2015	Workshop: ROS Community Workshop	
EO5	2015	Two days ROS training workshops for robot software developers	Workshop on Robot Operating System in Glasgow, UK
EO6	2015	A week summers school that entailed lectures in robot software and hardware and development	euRathlon/Sherpa summer school in field robotics in Oulu, Finland
EO7	2015	Two days conference on ROS software and robot software development (included in ROSCon recordings)	ROScon 2015 in Hamburg, Germany
EO8	2015	Partnering day for research projects funded by the European Union	euRobotics brokerage day in Brussels, Belgium
EO9	2016	Workshop: Robot Ontologies Workshop to discuss the modelling of robot structures	European Robotics Forum 2016 in Ljubljana, Slovenia
EO10	2016	Workshop: How Do We Surpass Current Barriers to Efficient Deployment of New Robotics in Industry?	
EO11	2017	Workshop: AI & Robotics: Delivering Platform and Integration Tools	European Robotics Forum 2017 in Edinburgh, Scotland
EO12	2017	Workshop: System Engineering - RobMoSys: the next level of a Model Driven Robotic Software Ecosystem	

18 Appendix I

Semi-structured interviews, each one of approximately 1 hour of duration.

Code	Year	Interviewee
IN1	2015	A researcher in a robotics laboratory
IN2	2015	A researcher in a robotics laboratory
IN3	2015	A business development manager in a robotics incubator
IN4	2016	The CEO of a small-medium sized robotics company