Mats Neovius

Trustworthy Context Dependency in Ubiquitous Systems

Turku Centre *for* Computer Science

TUCS Dissertations
No 151, November 2012

Trustworthy Context Dependency in Ubiquitous Systems

Mats Neovius

To be presented, with the permission of the Division of Natural Sciences and Technology at Åbo Akademi University, for public criticism in the Auditorium of *Gamma* on *November 26*th, *2012* at *12*:*00*.

Åbo Akademi University
Department of Information Technologies
Division of Natural Sciences and Technology
Joukahaisenkatu 3-5, 20520 Turku

2012

Supervised by

Professor	Kaisa Sere
Department	Department of Information Technologies
University	Åbo Akademi University
City, Country	Turku, Finland

Adjunct Professor	Luigia Petre
Department	Department of Information Technologies
University	Åbo Akademi University
City, Country	Turku, Finland

Reviewed by

Associate Professor	Christian Damsgaard Jensen
Department	Informatics & Mathematical Modelling
University	Technical University of Denmark
City, Country	Lyngby, Denmark

Doctor	Mauno Rönkkö
Department	Department of Environmental Science
University	University of Eastern Finland
City, Country	Kuopio, Finland

Opponent

Associate Professor	Christian Damsgaard Jensen
Department	Informatics & Mathematical Modelling
University	Technical University of Denmark
City, Country	Lyngby, Denmark

ISBN 978-952-12-2808-7

Abstract

The modern society is getting increasingly dependent on software applications. These run on processors, use memory and account for controlling functionalities that are often taken for granted. Typically, applications adjust the functionality in response to a certain context that is provided or derived from the informal environment with various qualities. To rigorously model the dependence of an application on a context, the details of the context are abstracted and the environment is assumed stable and fixed. However, in a context-aware ubiquitous computing environment populated by autonomous agents, a context and its quality parameters may change at any time. This raises the need to derive the current context and its qualities at runtime. It also implies that a context is never certain and may be subjective, issues captured by the context's quality parameter of experience-based trustworthiness.

Given this, the research question of this thesis is: *In what logical topology and by what means may context provided by autonomous agents be derived and formally modelled to serve the context-awareness requirements of an application?* This research question also stipulates that the context derivation needs to incorporate the quality of the context. In this thesis, we focus on the quality of context parameter of trustworthiness based on experiences having a level of certainty and referral experiences, thus making trustworthiness reputation based. Hence, in this thesis we seek a basis on which to reason and analyse the inherently inaccurate context derived by autonomous agents populating a ubiquitous computing environment in order to formally model context-awareness.

More specifically, the contribution of this thesis is threefold: (i) we propose a logical topology of context derivation and a method of calculating its trustworthiness, (ii) we provide a general model for storing experiences and (iii) we formalise the dependence between the logical topology of context derivation and its experience-based trustworthiness. These contributions enable abstraction of a context and its quality parameters to a Boolean decision at runtime that may be formally reasoned with. We employ the Action Systems framework for modelling this.

The thesis is a compendium of the author's scientific papers, which are republished in Part II. Part I introduces the field of research by providing the mending elements for the thesis to be a coherent introduction for addressing the research question. In Part I we also review a significant body of related literature in order to better illustrate our contributions to the research field.

Svensk Sammanfattning

Dagens samhälle är i allt högre grad beroende av programvara. Exekverbar programvara, kallat applikationer, körs av processorer, använder minne och svarar för kontroll och reglage av funktionalitet som ofta tas för given. Typiskt för en applikation är att den justerar funktionaliteten i respons till en viss *situation*. En sådan situation präglas av ett antal *kontext*. Varje kontext i sin tur förses eller härleds från inexakta givare, vilka gestaltar något informellt fenomen med varierande kvaliteter.

För att modellera en programvaras beroende av en situation bör dess kontext inexakthet approximeras. Detta förutsätter abstraktion och antaganden av omgivningen vilket följaktligen möjliggör rigorös modellering. Rigorös matematisk modellering förlitar sig dessvärre på atomisitet, dvs. en kontext uppdatering är förutsägbar. Rimligen är detta inte fallet för en funktionalitet med autonomt verksamma aktörer i ubikvitär datateknik, t.ex. på grund av mobilitet. Därför är den gällande kontexten och dess kvaliteter i vilken programvaran exekverar aldrig säker och kan vara subjektiv, vilka utgör ämnen för en kontexts kvalitetsparameter *tillförlitlighet*.

I denna avhandling, undersöks nivån på en kontexts kvalitetsparameter tillförlitlighet samt dess härledning i syfte att ge en klarare presentation av omgivningen åt programvaran. Tillförlitlighetsparametern identifierar en aktörs förväntningar på en kontext samt dess övriga kvalitetsparametrar. Nivån av tillförlitlighet fastställs av den kontext beroende aktören. Därmed fångar tillförlitlighet in eventuella fördomar och förväntningar samt är subjektiv givet ett kontext utfärdat av en aktör. Av detta följer behovet att behandla nivån av tillförlitlighetens (o)säkerhet.

Givet detta utformas forskningsfrågan som: *I vilken logisk topologi samt hur kan kontext utfärdat av autonoma källor härledas och modelleras formellt för att möta med en kontext medveten applikations krav?* Mer specifikt redogör denna avhandling för problemställningar gällande härledning av inexakt data i syftet att användas ändamålsenligt i programvara. I avhandlingen framställs en logisk topologi för kontext härledning, presenteras en generell modell för lagring av erfarenheter samt modelleras beronedeskap formellt inom Aktion System ramverket. Som en följd av detta studerar avhandlingen på vilket sätt det går att ändamålsenligt modellera och beräkna osäker information att presenteras åt en agent som är beroende av den vid körtid. Avhandlingen motiverar tagna beslut genom referenser till relaterad forskning.

Tekniskt sett är avhandlingen ett kompendium av vetenskapliga artiklar där skribenten medverkat, vilka är återpublicerade i Del II. Utöver introduktion av forsknings området i Del I, förser denna del nödvändiga element för att avhandlingen kunde förstås som en sammanhängande helhet, inklusive definition av en kontexts härledningstopologi som ett polyträd.

Acknowledgements

This thesis is a work highly influenced by, and made possible thanks to, its author's *contexts*. The inspiring contexts next to and in which I have been privileged to work in during my PhD studies have raised the quality level of this final version of the thesis. Of the whole thesis, this small chapter is the place where I may express my sincerest gratitude to the contexts characterising this thesis.

First of all, I owe my deepest gratitude to my supervisors, Professor Kaisa Sere and Docent Luigia Petre; they have been the most important contexts in my proximity. I am especially grateful to Kaisa Sere for the amazing (and rare I believe) ability of introducing a fresh student to topics that sparkles the student's interest; I admire her broad view on research. Moreover, I am thankful for all the time she took to discuss research with me, as well as for her patience in teaching me academic writing. I am also grateful to her for making it possible for me not worry about the financial arrangements of my PhD studies; I have been privileged to *depend* on her by an absolute level of *trust* in this issue. To my second supervisor, Luigia Petre, I am grateful for her continuous support. In times of "challenging moments", she has been the person that made me look on the positive side; I thank her very much for this because without such encouragement and coaching, this endeavour of mine might never have reached this point or quality.

Associate Professor Christian Damsgaard Jensen from Technical University of Denmark and Docent Mauno Rönkkö from University of Eastern Finland kindly agreed to review this thesis. Their careful review and constructive commenting greatly helped me to pull it all together into what it is today; I sincerely thank them both for this. In addition, I would like to especially thank Professor Jensen for accepting to be the opponent for the public defence of my thesis.

I am honoured to have been a student of the Turku Centre for Computer Science graduate school and am grateful for the financial and administrative support I have received. In addition, I am privileged to have been a part of the Department of Information Technologies at Åbo Akademi University and would like to express my gratitude to all the administrative personnel for their support. Moreover, I am sincerely grateful to all the persons in the context of the Distributed Systems Laboratory for making it such an inspiring and good working environment. Of these, my special thanks are directed towards my co-authors, Lu Yan, Manoranjan Satpathy, Pontus Boström, Ian Oliver, Marina Waldén, Fredrik Degerlund and Petter Sandvik; I have learned a great deal from working with them. I wish also to express my gratitude for the financial support in form of generous scholarships I have received from the Nokia Foundation, Hans Bang Stiftelsen and TOP-Säätiö. As the list of contexts to thank could go

on, I merely wish to express my thanks to all the other contexts that have been influential.

In addition, I take this opportunity to thank my friends for giving my life some extra meaning. Time spent with you is always well invested; there is only a handful of things better than, for example, a round of disc golf in the company of good friends.

Finally, last but definitely not least, I owe my gratitude towards my family. All the love I got, and keep on getting from my beloved wife Meri and the smile of the three most precious contexts in my life, Oscar, Elsa and Anni are irreplaceable. Without your love, none of this would have been achieved. After all, you are the contexts that make my life worthwhile.

30th October 2012, Turku, Finland

Mats Neovius

Contents

List of figures

List of tables

Part II: Publication reprints

Paper I: Mats Neovius, "*An Abstract Model for Incentive-Enhanced Trust in P2P Networks*". In: Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuanshun Dai, Laurence T. Yang (Eds.), *Embedded and Ubiquitous Computing - EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES, Nagasaki, Japan, December 6-9, 2005.* , Lecture Notes in Computer Science vol. 3823, 602 - 611, Springer Berlin / Heidelberg, 2005.

Paper II: Mats Neovius, Lu Yan, "*A Design Framework for Wireless Sensor Networks*". In: Khaldoun Al Agha (Ed.), *Ad-Hoc Networking: IFIP 19th World Computer Congress, TC-6, IFIP Interactive Conference on Ad-Hoc Networking, August 20-25, 2006, Santiago, Chile* , IFIP International Federation for Information Processing vol. 212, 119 - 127, Springer, 2006.

Paper III: Mats Neovius, Kaisa Sere, Lu Yan, Manoranjan Satpathy, "*A Formal Model of Context-Awareness and Context-Dependency*". In: Van Hung Dang, Pandya Paritosh (Eds.), *Proceedings of the fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)*, 177 - 185, IEEE Computer Society Press, 2006.

Paper IV: Mats Neovius, Kaisa Sere, "*Formal Modular Modelling of Context-Awareness*". In: Frank S. de Boer, Marcello M. Bonsangue, Eric Madelain (Eds.), *Formal Methods for Components and Objects, 7th International Symposium, FMCO 2008, Revised Lectures*, 102-118, Lecture Notes in Computer Science vol. 5751, 2008.

Paper V: Mats Neovius and Kaisa Sere. "*Mastering the Relevance of Subjective Information in Ubiquitous Computing*". Submitted to International Journal of Networked Computing and Advanced Information Management (IJNCM) Special issue on Social Informatics and COMputing (SICOM), 2012.

Part I:

"*Imagination is more important than knowledge.*" – Albert Einstein 1931

1 Introduction

In this chapter we provide an introduction to the concepts studied. We describe a general background on which the approach is motivated; we outline the research hypothesis, the research question and the adopted methodology. We briefly highlight the contributions and the limitations of scope as well as outline the rest of the thesis.

The notion of *context* is central in several disciplines [43] [46]. For instance, humans are very good at recognising, perceiving and adapting to the *implicit* context such as gestures, tone of voice, etc. This is called grounding [63] and implies that humans are innately context-aware [48]. Thus we understand context as some information that characterises the situation of entities, here humans. Being aware of this context and adapting to it may be considered a sign of intelligence, i.e. to be context-aware may be considered a characteristic of an intelligent entity. Contrary to humans, computers are very good at acquiring, aggregating, composing and processing data [90] by mathematical logical instructions. These instructions manifest themselves as applications. Inputs of an application are necessarily explicit, whereas the contexts are the implicit matters of informal origin.

An application that consumes and adapts to such context is context-aware. As the application provides a user means to perform a task [26] [27], an application is context-aware whenever it provides this means defined by contexts [238]. Hence, contexts sensed and derived in the environment of an application may rise to the level of a situation having an influence on the performance of the user initiated task. For example, a conference assistant user application may shift a phone's means of alarm between vibration and sound depending on the whether or not a presentation is attended. This is an example of a 'user application' consuming situation(s) that we distinguish from an 'application' that by consuming context provides derived context(s).

A user application task typically resolves some informal need of a user, with the help of some actuators. An actuator does, therefore, consume some formal event and produce an informal event manifesting the purpose of executing the user application. Dually, we recognise a sensor to capture an informal event that by an application provides a formal event which may further be used by other applications. Hence, stating that the beginning and end of each task is informal [275] is reasonable. This is motivated as the formal mode merely extends the informal mode, it does not replace it [198]. Thus, a formal specification with all its advantages in terms of expressing unambiguous matters applies only on an idealised view of the informal world [4], the model. Moreover, the coarser approximations on the modelled reality, the greater the risk of alignment errors in addition to discretisation errors. Therefore, context and context-awareness as unpredictable matters that describe the environment break down the purely algorithmic model of the formal mode demanded to show mathematical correctness [234] [235]. This motivates quality parameters of a context as a means to represent a model's relatedness with reality, i.e. in terms of Abrial [4], how far from the real environment the model is. Hence, a context as considered in this thesis is a digitalised representation of a continuous analogue real world phenomenon whose quality parameters capture the consequences of the discretisation errors as well as other alignment errors.

With this, we have no intentions of devaluing the importance of analysing properties of software in a formal mode for the sake of increasing behavioural

certainty and for having a structured means to reason on this. Our intentions are merely to stress that mathematical logical rigour and proofs may not imply correct system behaviour, i.e. that *correct*ness is a mathematical property. That is, in addition to serious challenges in defining correctness in engineering [202], a context-aware user application's behaviour is a realisation of a complex composition of inherently imperfect context to a situation. An empirical survey from industry studying machines (automatic assembly line, paper product line, forest harvester and rock drill) strengthens this point of view, showing that a majority of erroneous behaviour originate from human errors or wrong, slack, or loosened fitting of the context sensing devices [137]. Common to these matters are that they are outside the domain of mathematical correctness, i.e. of informal causes.

Informalities need, under certain assumptions, to be considered formally. These assumptions manifest the necessary axioms that if violated, something fundamental is very wrong and nothing else may be considered certain either, i.e. a formal model must be correct. For example, stating and trusting an apple tree not to bear cherries must be acceptable though philosophically even this could be argued [131]. In this example we refer to *trustworthiness* as the level of belief in this proposition that captures a level of arguable assumptions involved in the statement. For example, before bearing the first apple, i.e. in the context of a plant and not a tree, only given that the plant is accepted as an apple plant and not a cherry plant (which is not easy to tell) we may trust it to bear apples, if any, in the future. Consequently, the foremost assumptions for establishing necessary axioms demanded for analysis are that the input data is *perceived in context* and that it is *trustworthy*; concepts that make up this thesis. This, in addition to the other problem statements described above lead to the formulation of the research question that this thesis aims to shed light on:

> *In what logical topology and by what means may context provided by autonomous agents be derived and formally modelled to serve the context-awareness requirements of an application?*

The research question and methodology are further outlined in Section 1.3, while specific contributions and limitations of scope are discussed in Section 1.5.

1.1 Background

We have come a long way from vacuum tubes amplifying signals, the pioneering work of the transistor in 1947 by Bardeen and Brattain, and the integrated circuit in the 1950's by Dummer, Kilby and Noyce. All these contributed significantly to the electronics revolution. The integrated circuit is often considered the catalyst for the Information Age where one modern desktop computer's microprocessor contains thousands of millions of transistors. Later, being connected 'all the time everywhere' [181] transformed the Information Age into,

the so called Information Revolution with applications producing automated transactions [10].

In the early days of integrated circuits, the limited contextual availability and stationary / dedicated nature of the devices resulted in applications that were tailor made. Typically, these applications were expected to run in static environments [223]. This fostered mathematical modelling of applications. However, with the development of electro-mechanical devices, reduction in the size of transistors on the integrated circuit has combined with reduced energy consumption, production costs, increased mobility, and device connectivity; the means to realise the once fictionary deployment scenarios of computerised gadgetry have become reality.

This development first enabled the connectivity of stationary nodes, to distribute the workload. Such systems came to be known as distributed systems. Later mobile computing added mobility in the form of 'availability anywhere' to distributed systems, as depicted in Figure 1 inspired by [222] [239] [263]. Mobile computing also featured a degree of context-awareness, e.g. location awareness. Eventually, this development led to what is known as pervasive / ubiquitous computing [222] [251] [252]; the third wave of computing [239] [263].

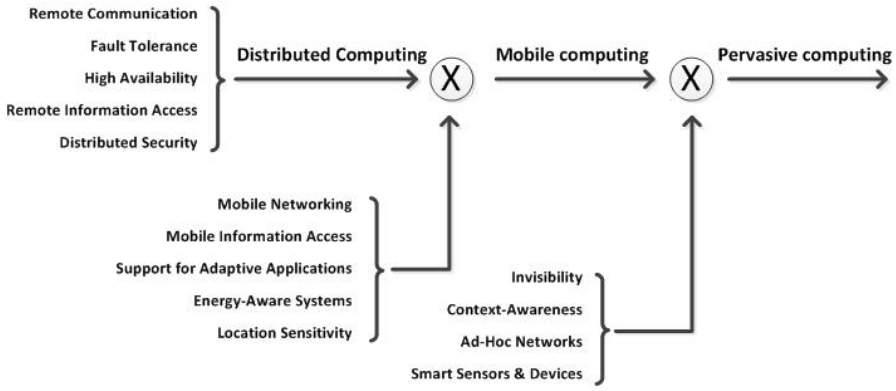

Figure 1 The evolution chain of computing

In ubiquitous computing, the technologies are being weaved indistinguishably to our everyday life, i.e. they disappear to the omnipresence as envisioned by Weiser [253]. The interface to a ubiquitous computing application is often transparent [3]. That is, when using a ubiquitous computing application the user may not be aware of this. Hence, some authors claim the term cloud computing to originate in ubiquitous computing [36]. Consequently, an ubiquitous computing user application provides a means for a user to perform a task with the device being a mere portal to the application space and the computing environment, the user's information enhanced physical environment [26] [27] [73]. Moreover, for a ubiquitous user application to be minimally

intrusive, it needs to be aware of the context it functions in [222]. The extreme of this view is the Internet as one computer, the pervasive cyberspace [7], reflecting the vision of computing becoming invisible, location and device independent with functionality accessible everywhere all the time [181].

The ubiquitous computing concept has given rise to study paradigms that build on it. Calm Technology considers how not to saturate a user with information [254]. In Calm Technology, the key is for information to migrate between a user's centre of focus and periphery, e.g. in car navigators the driving direction is not of interest before coming to an intersection when direction migrates to the centre of focus, for example, by voice guidance. Other related concepts include Ambient Intelligence [274] and Autonomic Computing [157]. Ambient Intelligence studies characteristics demanded by a ubiquitous computing environment in order to be intelligent and responsive to presence, e.g. sharing a virtual whiteboard only with students attending the lecture.

Autonomic Computing, on the other hand, considers how computers may eventually make decisions in favour of us. The vision is for the autonomic system to monitor the context, analyse it, construct plans and execute them based on the analysis in order to relieve humans from interacting with the system. Elements of the autonomic systems need therefore to self-configure, self-monitor, self-adapt and self-heal. Related to autonomic computing is autonomic communication that focuses on the self-* properties of the networks rather than computation [82]. From these, yet another concept called task computing [172] [183] has emerged. The focus of task computing is on a user's intents with respect to what resources are available.

All of these disciplines are context dependent. Common to all these post-centralised computing concepts is that they interact with one and each other in addition to adapt to the momentarily setting. Hence, all of them are context-aware.

1.2 Motivation

The amount of data created by the digital universe is estimated to increase from 487 ExaBytes ($487 * 10^{18}$) in 2008 to 5 fold in 2012 according to IDC's estimates [138]. With an increasing portion of this information being potentially available all the time everywhere, a ubiquitous computing dream with trillions of connected computing devices providing data outlines an environment in which navigation is of extreme complexity. In addition, this information availability has contributed to applications breaking loose from the confinement of a single agent observed at design time to Internet scale runtime environment [66] [181].

This new environment, in which computations are executed, is faced with issues regarding dynamicity and selection of relevant from irrelevant information. The promise of context-aware computing is to consider these issues [119]. Addressing them demands binding of context transparently at runtime

[48]. Simultaneously, the notions of trust and privacy policies between the context providing and consuming agents emerge because:

(i) Data collected from the personal ubiquitous devices is increasingly intimate [164] giving rise to policies abstracting the details irreversibly [27].

(ii) Acquired context's qualities need to be defined.

To address these issues, researchers have (i) considered the policies typically as logical rules evaluated by an agent in possession of the requested resource. These policies are local to the agent and mathematical logical analysis on the policies consistency is possible. On the other hand, acquired context's qualities (ii) are important due to the inherent imperfection of the context and autonomy of intermediate agents. This is noted as a main research issue when derived from uncertain contexts [270]. Together, (i) and (ii) constitute the motivation of this thesis.

1.3 The Setting of the Thesis

In one sentence, this thesis is concerned with finding a basis on which to reason and analyse inherently imperfect contexts that are derived by autonomous agents populating a ubiquitous computing environment. The imperfection stems from the inherent inaccuracy of capturing the informal environment. This is modelled by the quality parameters of a context. Therefore, coming to terms with such imperfection is necessary and providing a logic and defining an architecture is sought based on which to calculate with the quality parameters. Such architecture separates concerns between a part deriving context to a situation and a part consuming the situation and reacting to it logically.

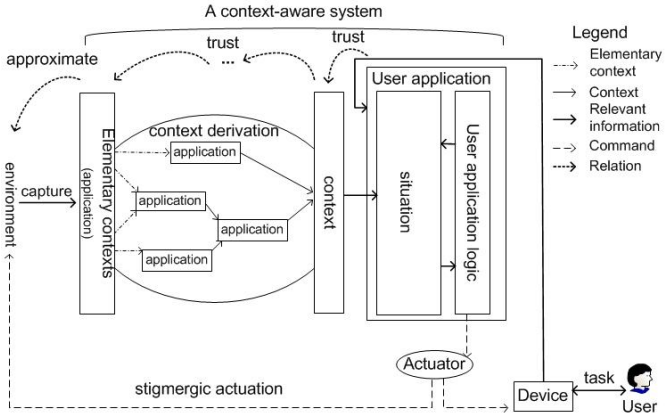

Figure 2: General context-aware system view

The setting that this thesis seeks to define and describe a context-aware system model as outlined in Figure 2. In Figure 2, the informal events of the informal environment are captured as *elementary contexts* that are the most basic form of context with a formal representation. This elementary context formal representation of an informal event may not capture all aspects of the informal environment, motivating a context's inherent imperfection [123] [126] and approximation relation in Figure 2. These inherent imperfections are captured as the metadata of a context, commonly called Quality of Context (QoC). The QoC is a set of parameters. Of the QoC parameters trustworthiness is considered in detail in this thesis, hence the trust relations. These QoC parameters propagate throughout the context derivation. Contexts are derived in applications to increase their level of information, denoted in Figure 2 as applications within context derivation. Such derived context is called contextual information. The logical topology of context derivation is defined in this thesis. Eventually the context including its propagated QoC parameters is consumed by a user application as a *situation*.

The user application evaluates the provided situation by the level of trustworthiness on the provider in this proposition. On this evaluation the user application applies a policy that determines to what extent the acquired situation influences the user application's logic. Hence, the user application logic that may trigger an actuator is influenced by a situation basing on imperfect context(s). The actuation, on the other hand, indirectly influences (stigmergically) the environment in accordance to the task, i.e. the actuation is stigmergic with respect to the contexts. Examples may include adjusting a valve controlling the air conditioning system or merely display the result on a display. This motivates the user application's logic in separation from the situation evaluating context applicability in Figure 2. Moreover, the device may provide the user application with commands. Hence, Figure 2 outlines a general view of the context-aware architecture we consider with separation between an application and a user application.

1.4 Research Question and Methodology

The challenges with respect to trustworthiness, context-awareness and a formal treatment of these are manifold. The hypothesis of the research that this thesis presents is:

> *In a network populated by collaborating autonomous context providing agents, it is possible to formally specify the contextual dependencies of context-aware user applications performing user-centric tasks in a scalable, maintainable and adaptable manner*.

This hypothesis calls for formally specifying a collaborating scalable and maintainable basis providing contexts to a context-aware user application.

Adapting to the current setting is crucial as the quality, availability and applicability of a context providing agent may vary. For example, a printer may run out of ink implying change of context. Moreover, as the environment of the context consumer may change, the applicability of a context varies, e.g. the closest printer is dependent on the location of the inquirer. These changes are subject to being aware of the momentarily context.

With this hypothesis, the research question this thesis aim to answer is:

> *In what logical topology and by what means may context provided by autonomous agents be derived and formally modelled to serve the context-awareness requirements of an application?*

This question yearns for elaborating on how to derive informal context to a situation that may support a user application's decision in providing a user means to perform a task. This is also the specific problem setting the research presented in the thesis.

The research methodology applied on this research question includes both exploratory as well as constructive aspects. The exploratory research relates to how the context's quality parameter of trustworthiness may be modelled and what its restrictions are. This yielded the confinements resulting in a polytree logical topology for context derivation as well as the necessity to acknowledge subjectivity and (un)certainty. Trustworthiness as a QoC parameter is noted by many related works [54] [74] [158] [176] [236] [237], but to the best of our knowledge, only examined in detail by Grossman et al. [113] whose approach supports ours. Constructive research methodology is adopted when formally modelling context and context propagation. Perhaps the most exemplifying of our constructive research is the formal dependence operator binding a context.

1.5 Contribution and Limitations of Scope

With respect to the setting of this thesis and the research question, the following challenges are addressed:

1. providing a means to include the ambiguous, unpredictable and uncontrollable context in a formal manner
2. introducing a scalable end-to-end model for context derivation
3. providing a model for calculating with the QoC parameter of trustworthiness

Thus, this thesis presents a formal means in which to model context and its dependencies (1). Our more specific contribution to this topic are provided in Papers III, IV and V defining the dependence operator in the Action System framework; also considered in Section 5.1. Challenge (2) calls for a model in which elementary contexts are derived to situations. For this, our contribution includes the componentised views of context presented in Papers II, III and IV. Moreover, in Section 3.5, a novel view on the context derivation's logical

topology is proposed, motivated and defined to be a polytree; which is a contribution in its own right. Challenge (3) considering the QoC parameters is addressed more specifically by studying the parameter of trustworthiness in detail. Our contribution to this challenge includes, to the best of our knowledge, a novel view of using the Subjective Logic framework on a general model of the recorded experiences for calculating the trustworthiness of context. Moreover, the Subjective Logic allows for ascertaining the level of trust by referrals on some proposition. Paper V proposes a means to consider as referrals only entities that share likes of the subjective matter they evaluate.

The scope considered by this thesis is limited to the QoC parameter of trustworthiness to how an application acquires its context, and to how contexts are derived. Hence, in this thesis we do not consider context discovery, ontology of contexts, artificial intelligence or context reasoning methods in a natural language. Moreover, we neither consider synchronisation of the sources, try to formalise the context as a construct, study types of information representation, weighing between the QoC parameters nor address how appreciation is distributed in case of many contributors. We do not differentiate between classes of context (internal, external, social, cognitive and so forth) due to their subjectivity. Our approach simply assumes the contexts to be available all the time everywhere, hence the ubiquitous computing concept in the title. We discard engineering problem settings, e.g. the sampling rate. We take a data-oriented view on acquiring the context, i.e. whether the contexts are stored on a server, or directly connected to, or acquired from some middleware is, out of the scope of this thesis too.

Computer science approaches on trustworthiness, trust policies, security, privacy and access control are each only briefly mentioned. The main focus is on experience-based trustworthiness as it captures the ever changing and ambiguous context by experiences. In line with related work on trustworthiness, how or by what preferences the experiences are derived is not considered. Moreover, we omit considering the consequences of breaching a context's trustworthiness; as failure management, fault tolerance and dependability issues branches to a separate field of research [167].

1.6 Structure of this Thesis

This thesis consists of two parts. Part I is three-fold. Each chapter in Part I begins with a short description what that chapter presents. Part II of the thesis consists solely of republished publications.

In Part I, we start by introducing the research addressed in this thesis. Context and context-awareness including quality parameters are defined and discussed in Chapter 2 followed by context models and its architecture in Chapter 3 and eventually, trustworthiness in Chapter 4. Chapters 2 and 4 are divided into introduction, problem analysis, state of the art with respect to the

challenges and finally, success criterions that the chapter in question raised and this thesis aims to shed light on. Chapter 3 follows the same structure with the difference that the final sub-section 3.5 provides a contribution, the logical context derivation topology of a polytree. This is, to the best of our knowledge, a novel approach considering context derivation in a logical topology of a polytree. Chapter 5 motivates how all these concepts fit together and describes the formal methodology of choice, the Action Systems framework. The Action Systems framework is later used to specify and reason on the complex structure.

Following these sections, in Chapter 6 we provide a short description of scientific publications with the author's role emphasised. In Chapter 7 we discuss our results, raising points of criticism and answering these. The discussion is followed by conclusions and future work in Chapter 8, a list of abbreviations and short term definitions in Chapter 0 and a list of referenced work in Chapter 10.

With the kind permission of the copyright holders, Part II of this thesis consists of republished publications of the author in accordance to Chapter 6.

"*For me context is the key - from that comes the understanding of everything.*" – Kenneth Noland

2 Context and Context-Awareness

In this chapter we define context and context-awareness in their various forms as used in this thesis. We also address the representation of context, including its quality parameters that capture a context's inherent inaccuracies. Moreover, we outline an application that may derive on a context. The structure follows that presented in Section 1.6.

In the natural language, ‘context’ merely consists of 7 characters in the Latin alphabet that when separated by spaces, is noted as a word. The definition of the word context in the Merriam Webster’s dictionary is: “the parts of a discourse that surround a word or passage and can throw light on its meaning” [188]. Hence, in the natural language, the context in which a word is written depends on the sentence; the context of the sentence on the paragraph; the paragraph on the book and the book on the definition provided. Consider the phrase “Sorry to hear that, but better luck next time.”. This is a grammatically correct phrase in the English language but the reader cannot perceive its true meaning without knowing its context: better luck to what? Why “but”? Sorry for what? [259]. Moreover, the event that triggered this sentience may depend on the history events [112], i.e. a sequence of events that led to this context. Consequently, knowing the context of an event provides a means for a better, more precise perception of the informalities at hand; these may be used to serve a user’s customised intents that contrary to humans, computers cannot yet, if ever, master very well. Having said this, we consider context as information shedding light on an entity’s informal environment of which the QoC parameters constitute the metadata capturing its imperfection.

At its simplest, context is captured by a sensor attached to a device executing a context-aware application that provides a situation to the user application. An application is context-aware whenever some context, and a user application whenever some situation respectively and their QoC influence it [38], i.e. being context-aware is to be responsive to the situation / context of the task [86] [168]. To be responsive may, or may not, trigger an update or an actuator. However, as both an application and a user application is implemented in a programming language that ideally is well-defined, the application and user application as a concept may be considered part of the formal mode of a task. Consequently, a context-aware application or user application is always triggered by a context, i.e. by some informal real world event. This makes all adaptive applications fundamentally context-aware [238] [276].

Such a view is supported by Zemanek who states that “no formalism makes any sense in itself; no formal structure has a meaning unless it is related to an informal environment” [275] where the informal environment may refer to context and the formal structure to an application. Naur [198] enforces this view by arguing that a formal mode extends the informal but does not replace it. He argues against the claim that “an expression in an informal mode can be conveyed by a formal expression” by that “the meaning of any expression in formal mode depends entirely on a context which can only be described informally” and continues stating that the meaning of the formal mode is introduced by means of informal statements [198]. That is, a formal proof based on facts often requires an intuitive understanding of these facts, hence, demanding passing between the formal and informal modes with ease, e.g. proving relation *descendent of* to be transitive requires an informal understanding of descendent and its difference to a similar relation of *child_of* .

These views set the approach of this thesis. If an application is entirely and natively context-unaware, then it cannot provide anything of interest to the informal environment, i.e. it cannot provide for a task of a user's interest. That is, if an application does not include informalities, then whatever it outputs is of reduced relevance and doubtful usefulness [198]. Dually, whenever an application is context-aware, it approximates some characteristics of the informal environment, i.e. contexts may never fully describe the current environment. Consequently, this section as well as the whole thesis set out to study matters related to deriving on informal context for supporting a user application's decision in means to provide a task.

2.1 Introduction to Context and Context-Awareness

The research on context and context-awareness originates from Olivetti's Active Badge research in 1992 [247] with the notion coined by Schilit et al. in 1994 [224]. Later research has split into a branch of Artificial Intelligence (AI) and natural language processing [33] [44] [47] [108]. Central questions in AI and natural language processing refer to the meaning of sentences as well as to methods to (dis)prove them together with follow up questions. For example, how to reason about the meaning of the sentence "is there water in the fridge" or how to (dis)prove "water in the fridge"; raising the follow up questions, how much water, in what form and during what period? Other directions within AI include context in information retrieval, in human-computer interaction and in distributed AI [43]. In addition, context has been considered in formal logics typically as an assertion [23] or basic assumption outlining a model's static part [4]. Perhaps because of these diverse views there is no commonly agreed definition on context, on what it is, what it entails and by whom / what it is created [32] [65] [72].

As a consequence, context has been defined in a number of ways [50] [60] [136] [170] [203] [218] [224] [248] [264]. In this thesis we adopt a frequently used definition in accordance to that of Dey and Abowd [76] considering context to be information that characterises the situation of an entity. According to Winograd [259], however, this definition is so broad that it covers nearly everything, from the electric grid to file systems. Having an application's view, Winograd [259] further stresses that something is context due to the way it is used in interpretation, not because of its inherent properties. Winograd's view could thus be put forward by the following example. In the context of *speeding* characterising an entity car, the information *temperature* is not a context. Obviously, *temperature* may be context for another setting. Hence, context does not exist by itself, but is used to describe an entity [84].

We consider information to possibly be context regardless of its instantaneous relevance to an application's event as it may become relevant at

some later point and must therefore, not be neglected. That is, information as a part of what led to the current context is context in its own right, e.g. the context of a book's loan period may be irrelevant until overdue. Hence, our view is related to the AI view that considers a context (situation) as "a finite sequence of actions. Period. It's not a state, it's not a snapshot, it's a history" [212]. This view defines an axiom stating that executing *action* in *context* is equal to executing *action*' in *context*' if and only if *action* = *action*' and *context* = *context*'. Such an AI view is in contradiction to the state-based formal view considering an instantaneous state of the system in which, for example, a predicate transformer's total function may execute [23].

Related to the formal and AI view, McCarthy and Hayes [185] consider a situation as "the complete state of the universe at an instance of time". They correctly notice this to be impossible to capture and restrict themselves to only provide facts about a specific view, i.e. a partial situation. Moreover, Ghidini and Giunchiglia [104] note that within a partial situation, an observer is able to view everything.

The temptation to approximate context of a partial view to a complete view ignoring or assuming inaccuracies of it comes from the power of mathematical functions [170]. This underlines the need to approximate the context unambiguously to a model in order to formally analyse it [4]. Obviously, the level of approximations and assumptions define the model's validity on reality. This is the reason why a formal method is applied on a model; whose 'closeness' to the real environment is critical [4]. Abrial [4] also notes a fundamental issue in terms of context; that "it is quite clear that these elements cannot be formalized completely" (*sic*) [4]. In addition, this constitutes the motivation for context in the first place, where ambiguities are captured as quality parameters and provided to the formal model of a user application.

Having presented these quite varying views on context, we continue by presenting our definition of context used throughout the thesis.

2.1.1 Definitions for Context and Context-Awareness

What is considered context to an application depends on its boundary. When considering locally attached sensors providing information to an application, the context-aware system boundary is obvious and sharp; it features the sensors and the application. However, for distributed applications that interact and include remote procedure calls, the boundary gets blurred [170], i.e. should a remote procedure be considered context? In this thesis, however, we define the context boundary of an application to be sharp: all information used within an application but derived from outside is considered context, regardless its origin. Moreover, as we define context on a general level and not for a specific purpose,

the definitions of context and context-awareness are intentionally vague. This vagueness is the motivation for further categorisation of context in Section 2.1.2

Considering context this way, our definition of context follow Dey's and Abowd's [76] but include 'virtual objects' and setting:

Definition 1. Context: "*Context is any information that can be used to characterise the situation of entities. An entity is a person, place, object, virtual object or setting that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*"

Examples of person, place and object entities are *Alice*, *cafeteria* and *car* respectively, e.g. context *height* characterises the entity person, a *location* characterise a place and *next_to* characterises an object. 'Virtual objects' are entities that exist virtually, e.g. *board_of_directors*, *e-calendar* and *service* whose contexts may be *in_meeting*, *entry* and *available* respectively. The setting entity refers to relation properties on the entities characterised by contexts [86], playing a role to establishing context [122], e.g. settings *next_to* and *married* where *married* may be identified by contexts *time* and *spouse*. Moreover, with respect to the definition, the "*any information*" and "*characterise the situation*" suggests that all information used to characterise a situation of an entity or group of entities is, in its own right context. This includes social matters [133] [191] [205] [216]. This definition does not explain what the "*situation of entities*" is but illustrates it through simple examples [120]. This underlines the broadness of this definition of context [259], making it an umbrella concept allowing entities to be context characterising other entities, e.g. entities in proximity may be context.

To restrict the definition of context slightly, we note that a context, as used in this thesis, is sensed or derived from the informal environment, i.e. context is not a formal quantity that the application may control directly. Hence, a queue's length by image recognition is context whereas the state of the ticket dispenser is not. Thus, examples of contexts are: identity, spatial (location, altitude, speed), temporal (date, time, season), environmental (luminosity, humidity, temperature), social (close, reachable), resources (connected, availability), physical (blood pressure, area, thickness), activity (walking, sitting) [8]. Of these, for example the identity may not be sensed but provided by informal means. Examples of necessary matters for an application that are not context include variable, constants and state.

Having defined context, we consider an agent (used as a general term for application, user application or informal entity) context-aware if it consumes context or situation for deciding how, if at all to adapt. Hence, context-awareness is related to adaptivity, making all adaptive applications context-aware [75] [76] [238] [276]. Moreover, the context consumed may change at any point of time, e.g. as a consequence of the entity's mobility. Thus, the relevant context is a property of the moment and very hard to approximate and define at

design time [86] as it is defined with respect to the process [65] [86]. Consequently, we employ the following definition on context-awareness as:

Definition 2. Context-aware: "*An agent is context-aware whenever it adapts its behaviour / output according to the momentarily context.*"

The key of this definition is in the behaviour / output. We consider an agent context-aware if it combines contexts, calculates on acquired context or performs an actuation, e.g. computes speed from revolutions and circumference, calculates average or writes an entry to a log file. Consequently, the definition considers context-awareness per se, not by its direct relevance to the user, e.g. an entry in a non-rewritable log (earlier updated based on context information) makes the application editing the log file a context-aware user application as this entry may later become 'relevant' context. This definition of a context-aware agent excludes mere forwarding of a context, as a forwarding agent does not adapt, i.e. it functions in the same way regardless of the context. However, as something is context due to the way it is used in interpretation, not because of its inherent properties [259], a context-unaware forwarding agent may provide context information.

2.1.2 Categories of Contexts

Two disjoint categories of context may be recognised based on the means of acquiring the context. These are called implicit and explicit contexts and we define them as follows:

Definition 3. Implicit context: "*An implicit context is ambiguous information describing the environment."*

Definition 4. Explicit context: "*An explicit context is unambiguous command inputs."*

We further categorise sensors capturing the implicit context into physical sensors (e.g. temperature, humidity, location) and logical sensors (e.g. role, time) [156] [226]. Our categorisation relates to external and internal sensors [118] [119] [171] [209] [238] where the external context (physical sensors) provides a user's environment and the internal context (logical sensors) provides a user's internal state, e.g. cognitive (*next_to*, *busy*) or physical state (*position*). This distinction is, however, not always clear as for example, a user's social environment can be provided partially by internal and partially by external sensors [119]. Explicit context, on the other hand, captures information that is provided unambiguously, sometimes called control input, e.g. a command through a keyboard. Common to both categories is that they are sudden, i.e. they may not be anticipated in a clear and unambiguous manner.

With respect to the implicit contexts, it is notable that terms in categorisation vary. For example, Indulska and Sutton [139] categorised location sensors into

three types, namely physical, virtual and logical. They distinguish between these by means of capture, i.e. physical refers to GPS, virtual to determine an agent's location by time and calendar entry, whereas a logical sensor may determine the position by login at a desktop computer and fetching this computer's location from a database. Baldauf et al. [25] follow this three-fold categorisation but on a general level, not mentioning explicitly this to apply on means of sensing location.

In addition, we distinguish between two categories of implicit contexts: elementary context and contextual information.

Definition 5. Elementary context: "*An elementary context is unprocessed raw data captured by sensors."*

Definition 6. Contextual information: "*Contextual information is information that is derived from elementary contexts and other contextual information.*"

The elementary context (or context-primitive) relates to atom, direct, physical, source, provider, intrinsic whereas the contextual information (compositional context) higher level context, indirect, logical, context information, virtual context, context synthesisers output and situation respectively [25] [48] [66] [85] [103] [117] [130] [139] [211] [245] [271]. The term context is used when it is not important whether implicit elementary context or context information is meant. Moreover, we consider a key stroke to be an elementary context captured by the membrane switch. Hence, an elementary context is the product of an application that transforms an informal event captured by a sensor to a formal representation, in line with Figure 2. Characterising for such an elementary context is that it is independent of other context. Moreover, the elementary context, the contextual information and all their derivatives have no sense of temporality in their own right. Thus, a context is a snapshot at a certain moment whose sampling rate is sufficient, that when time stamped and stored is assigned a temporal aspect.

For example, a spatial elementary context of an entity is *location* where a sensor deriving *location* is attached to the entity, say a *mobile phone* used by *Alice*. Another entity, *Bob*, may share the same spatial elementary context '*location*' when associated with an entity whose location is known, e.g. *Bob* share *Alice's location* when associated by *in_close_proximity*. However, as *Bob*'s location depends on the relation between *Bob* and *Alice's mobile phone*, it is derived and thereof, contextual information. Contextual information *Bob*'s *location* is derived in an application from elementary contexts and/or other contextual information ascertaining the *in_close_proximity* relation. Hence, context is derived hierarchically. Altogether, this resembles the simple logics imposed by widgets built on top of widgets in Dey's Context Toolkit [77] where the widgets provide contextual information, Loke's Prolog style of rule relations [174], Henricksen et al. context modelling language [120] [126] to mention a

few. Issues relating to modelling of context are considered in greater detail in Chapter 3.

If the context rises to the level of being consumed by a user application that may, or may not, trigger an actuator based on this, the context manifests a *situation*. Thus, a situation derived from hierarchically organised contexts is a meta-level concept of contexts [200]. A situation is a prefabricated abstraction defining logical conditions on the constants and contexts [75] [84] [122] that we define as follows:

Definition 7. Situation: "*A situation is a specific configuration of context(s) and constant(s) consumed by a context-aware user application.*"

Consequently, we share the view on a situation with [65] [67] [68] [70] [71] in that all situations and contextual information derive from the same set of contexts. The fundamental difference between a situation and a context is that a situation is consumed by a context-aware user application, whereas contexts are consumed by a context-aware application. Consequently, a situation may be considered a wrapper abstracting the internal configuration of context from the context-aware user application, said to be "the semantic interpretations of context" [268]. Moreover, the set of all situations acquirable by a user application provides the partial view of the environment, the application's domain of discourse [105]; a matter elaborated on in Section 2.1.3.

A situation has internal and external perspectives [48] [83], called a 'context driver' by Lei et al. [169] and cascading context by Prekop [209]. This implies that a single context may contribute to several contexts (situations) [238]. Moreover, a context for some application may simultaneously be a situation to another user application [84] [85]. Hence, the way a context is used determines whether it is context or a situation. This topology is elaborated on in Section 3.3.1.

2.1.3 Context Derivation

According to Dey, "one of the main reasons why context is not used more often in applications is that there is no common way to acquire and handle context" [74]. He further notes that context handling is in general improvised, where application developers choose an implementation technique at the cost of generality and reuse. Partly as of this, this section outlines a general structure of context derivation. In this outline we follow the notions of Coutaz and Rey [66] in order to reason in a structured manner on context and its appearances with sharp boundaries on applications. This view concurs with the idea of separation of concerns between agents deriving context and agents consuming context, a matter further elaborated on in Section 2.1.4 and 3.1.1.

For this outline, consider the (unrealistic) set of gross context $C_G(t)$ to be the history of all observed facts together with those demanded by the user

application for providing the tasks at logical time *t*. Let the contexts observed by a context sensitive system at time *t* be $context_S(t)$, where subscript $_S$ stands for 'system'. Let $C_S(t)$ define the history of these observed contexts, i.e. $context_S(t) \subseteq C_S(t)$, $C_S(t) \subseteq C_G(t)$ and $C_S(t) = context_S(t) \cup C_S(t\text{-}1)$. Similarly, let the situations a user application is concerned with at a specific logical time *t* be $situation_A(t)$, where the subscript $_A$ for 'application' in user application, with $S_A(t)$ denoting the history of these, $S_A(t) \subseteq C_G(t)$ and $S_A(t) = situation_A(t) \cup S_A(t\text{-}1)$. The history of situations consumed by a user application is similarly defined as $situation_N(t)$ and $S_N(t)$, where the subscript $_N$ stands for 'net' as in net situations. In line, we have that $S_N(t) = S_A(t) \cap C_S(t)$ and $S_N(t) = situation_N(t) \cup S_N(t\text{-}1)$. Hence, our approach to the relation between contexts and situations including their histories are as follows:

$t = 0$: $C_G(0) \cup C_S(0) \cup S_A(0) \cup S_N(0) = \emptyset$

$t \geq 0$: $situation_N(t) = context_S(t') \cap situation_A(t)$

$t \geq 0$: $context_G(t) = context_S(t') \cup situation_A(t) \cup$ <*other observables*>

$t > 0$: $C_S(t) = context_S(t') \cup C_S(t\text{-}1)$

$t > 0$: $C_G(t) = context_G(t) \cup C_G(t\text{-}1)$

$t > 0$: $S_i(t) = situation_i(t) \cup S_i(t\text{-}1)$ for $i \in \{A, N\}$

A feature of this representation is that despite temporalities, $situation_A(t)$ does not need to consume the most recent $context_S(t)$, a feature well motivated when, for example, calculating the trend, or the average temperature during the last week. We note that an application needs to be able to demand 'old' context, as we do not consider a specific implementation. The relations are illustrated in Figure 3.

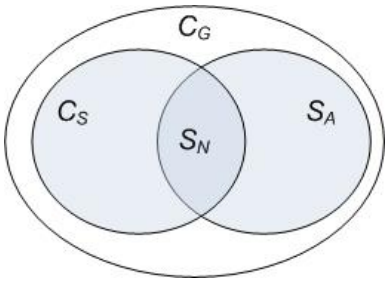

Figure 3: Situations and contexts

With this, the underlying system's context conformity with respect to the user application's desires is captured by $|S_N(t)|$. If $|S_N(t)| = 0$, the user application has not been affected by context(s) until time *t*. If $|S_A(t)| = 0$, the user application has been context-unaware until time *t*. Moreover, $|C_S(t)|$ compared to $|C_G(t)|$ denotes the whole system's intrinsic context sensitivity up until time *t*.

From a ubiquitous computing view, the removal or abstraction of outdated context is not an issue and all observed context are, for now, considered available. Hence, we consider an application that provides a context to store the

history of it. Moreover, this view of system contexts $context_S(t)$ and user application situation $situation_A(t)$ is an initial suggestion to the separation of concern in the context-aware architecture.

2.1.4 A Context-Aware Architecture

A starting point of division of a context-aware architecture consists in the separation of concern between user application $situation_A(t)$ including $S_A(t)$ and the context deriving applications providing $context_S(t)$ and $C_S(t)$. This is depicted in Figure 4 that is related to Figure 3 and Figure 2, i.e. the different levels of contexts and situations are noted in terms with concepts introduced in Section 2.1.3. Such separation of concern is a fundamental feature of any context-aware system for the sake of reusability and maintainability [8] [34] [56] [59] [60] [74] [76] [77] [83] [111] [165] [216] [227]. Baldauf et al. [25] state that this separation is the main criterion for a context-aware architecture. Moreover, it makes the user application's situation maintenance transparent, i.e. sufficiently abstract to free the context-aware user application from reasoning on the operational details but sufficiently precise for autonomous determination of current context [216].

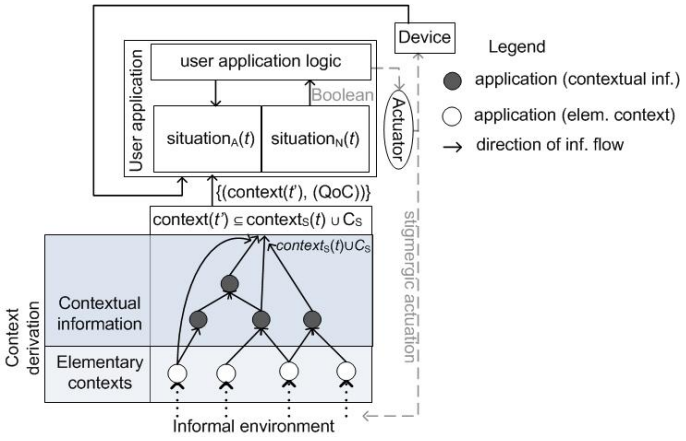

Figure 4: Schematic view of context derivation

Hence, for a user application to act on a non-empty set of contexts $context(t') \subseteq C_S(t)$ provided by autonomous applications, this context is defined in $situation_A(t)$ and becomes an element of $situation_N(t)$. A predicate deciding whether or not the user application is in context is applied on $situation_N(t)$, called exploitation [65] and context management layer [121] in related research.

Each $context_S(t)$ is provided by an application. Such autonomous applications share many features with an encapsulated component. Traditionally a component is defined to be a "unit of composition with contractually specified interfaces

and explicit context dependencies only" that is "deployed independently and is subject to composition by third parties" [241] [242]. Hence, key characteristics of any application (component) in context derivation are [48] [250]:

(i) *explicit dependencies* that specify the contexts the application requires in order to provide for its task
(ii) *contract and interfaces* specifying the functional and non-functional characteristics of the component, i.e. what is needed and what is guaranteed typically with pre- and postconditions
(iii) *unit of deployment* meaning that the component is an autonomous element that may interact with other components through its interface and;
(iv) *third-party composability* stating that the component may be further composed

Moreover, the application providing a context adheres to Szyperski's [241] [242] claim that a component has no externally observable state. This supports the independence of a context providing application.

Thus, we model an application deriving context by a uniform structure dividing the application internally to three parts: one acquiring context, one processing the acquired context and one providing the output [66] [97] [111]. We call these parts of a context *acquirer*, *application body* and *provider* and define them as follows:

Definition 8. Context acquirer: "*acquires the context(s) the application depends on.*"

Definition 9. Application body: "*conducts some algorithmic functionality on the acquired context(s).*"

Definition 10. Context provider: "*provides the output of the application.*"

With respect to a component's key characteristics, a context acquirer acquires contexts and defines the explicit dependencies (i) and the input interface (ii). It may also implement some selective predicate on the acquired contexts defining the means of context binding, e.g. a threshold. An application body executes instruction(s) on the acquired contexts whereas a context provider provides the new, improved contexts (ii) $context'(t) \subseteq context_S(t)$. Applications providing context may depend on other applications providing context (iii, iv) abstracting the $context_S(t)$ making the context derivation hierarchical.

We consider an application to have two different kinds of input and output: control and data. The control in / out constitutes a channel for unambiguous information, the explicit contexts, e.g. commands, inquiries, handshaking. Inputs on this channel influence the processing of the application. The data in / out consists of $context_S(t)$ including the QoC metadata. The QoC parameters are elaborated on in Section 2.1.6. Altogether, four different types of applications may be outlined: (i) an application providing elementary context $x(t) \in context_S(t)$, (ii) an application deriving context acquiring $y(t) \subseteq context_S(t)$ and

providing $z(t) \subseteq context_S(t)$ where $z(t) \cap y(t) = \emptyset$, (iii) an user application acquiring $\alpha \subseteq context_S(t')$ where $context_S(t') \subseteq C_S(t)$ and $context_S(t') \subseteq situation_A(t)$, as well as the (iv) *actuator* consuming 'control out' of the user application and stigmergically affecting the informal environment. These are depicted in Figure 5 that is influenced by related work, the component model [66] [214] and context handling component model [111]. This approach share the idea with sentient object model [35] [97] and the event-control-action pattern [83].

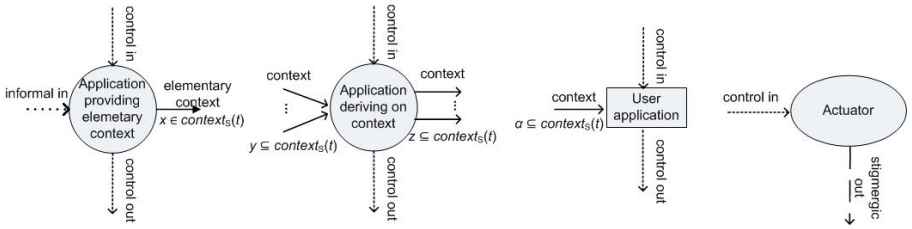

Figure 5: Types of applications

2.1.5 Context Acquisition and Modes of Adaption

According to Brown and Jones [51], there are two modes for context acquisition: *proactive* and *interactive*. These are called push and pull by Cheverst et al. [61] and synchronous stream of data and asynchronous events by Crowley et al. [71]. Proactive context acquisition refers to automatic acquiring of context with the context continuously available for processing at a given quality whilst interactive context acquisition update the context only on request.

Dually to proactive and interactive context acquisition, modes of adaption are either *active* or *passive* [60]. Active adaption refers to an application or user application automatically adapting in response to a context without user interaction. Active adaption is also referred to as automatic execution of a service [76], context triggered action [224] and contextual adaption [203]. Consequently, the design of a means for active context-awareness is delicate, as a user's intents are crucial to capture [222]. Passive adaption, makes the relevant context available for later retrieval or presents it to a human user for specifying how, if at all, to adapt; sometimes referred to as tagging [76], proximate combination and contextual commands [224]. Whether passive or active adaption is used is determined by the consuming application [60], implying that a single context may be part of both active and passive adaption. Examples of active and passive context-awareness are an automatic air conditioning and a web site without auto-refresh respectively.

Erickson [90] argues against passive adaption as it violates the purpose of context-awareness of letting the systems take actions autonomously. This desire

is facilitated by the fact that a user does not want to be in a control loop saturated by simple inquiries, the idea that gave rise to calm technology. However, active context-awareness will, most likely, never match the level of human context-awareness motivating in favour of passive adaption. Obviously, a mixed mode of adaption is possible, e.g. a control system may implement active adaption that shifts to passive and freezes the system if 'emergency stop' is pressed.

2.1.6 Quality of Context

An implicit context is derived from the informal physical environment. As the true configuration of the environment may not be accurately captured, a context is considered inherently imperfect [123] [126]. For elementary context, it may be *incorrect* when it fails to capture the true configuration of what it sheds light on, *inconsistent* if it is derived from non-unanimous information or *incomplete* if some aspect is abstracted or unknown. These inherent inaccuracies and ambiguities on context give rise to the concept of quality of context (QoC) and its parameters [54] [121] [158] [176] [205] [236] [237] [257]. We consider QoC as the metadata of context.

QoC is typically modelled as a set of parameters. The most important QoC parameters are according to Buchholz et al. [54]: *precision*, *probability of correctness*, *trustworthiness*, *resolution* and *up-to-dateness*. Here, precision refers to the relatedness with reality e.g. GPS accuracy; probability of correctness refers to the unintentional erroneous metric of the elementary contexts, e.g. frequency of internal errors typically acquired by testing; trustworthiness the rated certainty of the provider with respect to the other QoC parameters; resolution refers to the granularity of information, e.g. temperature inside may vary; and up-to-dateness refers to the age of the context.

The QoC parameter of trustworthiness is noted as a complex parameter [54] [236] and an interesting and open question by Dey [74]. Research referring to Buchholz et al. [54] does typically not include trustworthiness [257], evades considering it more closely [158] [236] [237], or simplifies the meaning to considering it as a specific instance [176]. To the best of our knowledge, Grossman et al. [113] are the first to consider means to calculate with trustworthiness as a parameter of QoC. They model trust as a triple (belief, disbelief, ignorance) but use, as stated, a simplified version assuming non-existent disbelief making belief behave alike a percentage of truth.

A feature of trustworthiness as defined by Buchholz et al. [54] is that it is the only QoC parameter that is interpreted by the agent consuming the context. As this context consumer cannot have any data by which to place a level of trust on a provider, trust needs to build up by experiences and includes the context consuming agent's expectations and cognition. It is also the only parameter that captures the complete performance of the provider including the other QoC parameters. The type of trustworthiness presented by Buchholz et al. [54]

therefore builds up from initial vacuous trust. Altogether, this makes the QoC parameter of trustworthiness experience-based and subjective, issues that are further discussed in Chapter 4.

In addition to these, Sheikh et al. [237] split the QoC parameter of resolution to *spatial* and *temporal* resolution. They stress that the spatial resolution describes the physical area (space) to which a context is applicable, i.e. a temperature may be applicable ± 5m. Dually, the temporal resolution describes temporal granularity, i.e. the time for which a context is applicable. As the temporal granularity varies, it implies that the rate of aging is not uniform [225], e.g. context *name* ought to age slower than *temperature*. Whenever the temporal granularity is modelled as a continuous function on a continuous datum, such as time (aging), with the granularity defined by a threshold on certainty, the context's certainty continuously changes. To the best of our knowledge, McCarthy [184] was the first to acknowledge this. Research on presenting and evaluating the QoC parameters as parameters that influence the 'worth' of the context is scarce, with Manzoor et al. [176] claiming to be the first to consider QoC parameters as the worth of context for an application.

There are several related concepts of QoC such as Quality of Service (QoS) and Quality of Device (QoD). A main difference between QoC and QoS is that QoC may exist without a service or a device, i.e. QoC is something related to data whilst QoS to the providing service. Moreover, QoD limits QoS and QoC to the hardware's capabilities [54]. The concept Quality of Information (QoI) is related to QoC and they are sometimes used interchangeably [257]. The relatedness is obvious also for the parameters of QoI. For example, a study of surveillance systems identified the following QoI parameters: *certainty*, *accuracy*, *integrity* and *timeliness*; where certainty, accuracy and timeliness surely overlap with the QoC parameters [135]. However, in this thesis we make a clear distinction: as information may be any data including context, we consider only context as some inherently imperfect data describing the environment. Hence, this thesis focuses solely on QoC.

2.2 Problem Analysis

The main challenge with respect to context stems from the difficulty to define the concept itself as well as what it describes. For example, a context's inherent inaccuracy breaks down the algorithmic model of an execution. Conversely, as the execution is formal, the contexts are precise in computation. Here, the contexts' metadata of QoC parameters come to play a decisive role in propagating the uncertainties related to a context, leading us to state Challenge 1:

Challenge 1 Define computations on an inherently imperfect context.

Challenge 1 basically calls for discovering functions to compute on contexts in order to algorithmically derive contextual information. Hence, a best-effort

context derivation ascertains not to introduce additional inaccuracy / ambiguity. In addition to problems related to representing the inherent inaccuracy of context, selecting the most suitable context providers for deriving an output with as high quality QoC parameters as possible is desired. Of the QoC parameters, in this thesis we focus on the parameter of trustworthiness, elaborated on in Chapters 4 and 5.

Ranking the possible context providers with respect to a parameter or configuration of continuously changing parameters provides a basis for straightforward dynamic binding. This requires rigorous and dynamic runtime binding of context providers, leading to Challenge 2:

Challenge 2 Model runtime binding of context applications based on QoC and suitability.

Providing a means to address Challenge 2 is crucial for context-awareness in an ever changing environment, as the whole concept relies on adaption to context. Together, Challenges 1 and 2 seek for a methodology in which to reason on the QoC parameters. An architecture supporting these matters is presented in Chapter 3.

2.3 State of the Art

Research on context and context-awareness as concepts is limited. Perhaps this is because they are matters of definition. At the time of writing, existing implementations on context and context-awareness are often restricted to the use and test of physical sensors providing factual metrics as context providers [25] [133]. In these, contexts are often assumed perfect [123] as opposed to imperfect. This view gives rise to gathering as much context as possible to serve for the ever finer grained contextual predicate of an application with the drawback of increasing complexity. The research focus has therefore shifted to architectural research on how to abstract, represent and identify relevant context (situations) that the user application is in need of from elementary context [269]. Hence, much of the state of the art on context and context-awareness is on considering the context deriving application, representation and means of binding the context; issues that we address in this section.

2.3.1 Context Processing Components

The contextor component [66] [214], the context handling component [111] and the sentient object [35] [97] are results of research on means to decompose a context-aware architecture to manageable elements. The main difference among these approaches is their point of focus. The focus of a context handling component and of a contextor component is on the communicational channels

noting that a context's metadata constitute the QoC parameters. A sentient object focuses on the algorithmic core.

In any of these models, when several of their elements (components or objects) are in succession, a hierarchy of a directed graph is formed. In the directed graph, *data in* channels of a more abstract element are connected to compliant *data out* channels of the more specific elements; and *control out* of the abstract to *control in* of the concrete element [35] [66] [97] [111]. Coutaz et al. [66] call this hierarchy a colony of components whose data flow they note to be static (design time), semi-static (run time at system launch) or transient (dynamically changing).

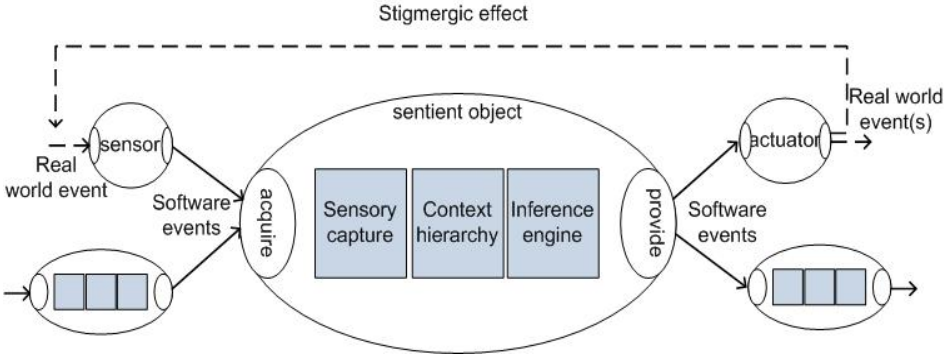

Figure 6: The sentient object model

In contrast to a contextor and a context handling component, a sentient object [35] [97], depicted in Figure 6, focuses on the internal functionality of the object consuming and producing software events. Three objects are defined in this model: *sensor* that consumes real life events and produces software events, *actuator* that consumes software events and produces real life events and the *sentient object* that consumes and produces software events. In the model outline, the actuators may influence the sensors stigmergically, i.e., by real life events such as by adjusting a valve; this may indirectly affect the sensor. Internally a sentient object is three phased: (i) sensory capture that performs acquiring and fusion of input events integrated as a Bayesian network to model the inaccuracy and dependency of sensor data, (ii) context representation / hierarchy that transforms captured and fused exclusive and exhaustive contexts (software events) to other context(s) in a hierarchy. These transformed representations are consumed by (iii) inference engine that reason by a set of rules to produce an output as a software event. Hence, a sentient object derives acquired *context* $X \subset C_S(t)$ to establish and provide a new *context* X', where $X \cap X' = \emptyset$. Moreover, in accordance to Figure 6 this software event may be consumed by another sentient object forming a hierarchy of objects. However, the authors do note that "essentially, a sentient object is an encapsulated entity, with its interfaces being sensors and actuators" (*sic*) [35] [97].

Consequently, there is a high degree of similarity between the sentient object and the application types presented in Section 2.1.4.The similarity of the application types in Section 2.1.4 with the contextor component [66] [214] and the context handling component [111] are also obvious. Hence, we omit presenting them in greater detail and direct interested readers to referenced work [66] [111] [214]. We note that a contextor component, a context handling component and the sentient object all encapsulate the functional update [48]. As the functional update is algorithmic, it may be formally modelled, guaranteeing not to introduce additional ambiguity. The informal updates are captured as changes in the input event / data or its metadata.

2.3.2 Context Representation

A context may be represented as a symbolic, a factual or a truth value, e.g. location as 'A5050', temperature as '293.15°' or standing '*true*' as a Boolean. Each of these representations have their own characteristics; the symbolic model refers to abstract symbols [95], e.g. staircase *A* and room number 5050, whereas the factual value is a specification of the context it describes, e.g. Kelvin scale, and the Boolean is an irreversible interpretation. In this section we present state of the art means of contexts representation and how this may be utilised.

Considering the context as a term in a first order predicate logic, Gu et al. [117] represent a context with the basic form *Predicate* (*subject*, *value*). With respect to our definition on context, we consider the subject as the entity. Hence, *Predicate* $\in V$ where V = {'predicate names'}, e.g. *location*, *status*; *subject* $\in S$ where S = {'entities'} and *value* $\in O$ where O = {'all possible values of S'}, e.g. O = {open, warm}. As predicates are Boolean valued functions, representation as a predicate is defined as $subject\mathrm{R}_{predicate}value \rightarrow Bool$. The expressivity of context as a predicate is limited to irreversible interpretation on a subject in a statement. Obviously, many such predicates may be combined by operations of the Boolean algebra. Hence, expressing transitive properties to model relations between concepts is possible, i.e. model an ontology [117] [246]. Concerns with respect to ontology are, however, out of the scope of this thesis.

When representing context by the dimensions that makes it up leads to context as a point in a three-dimensional space spanned by self, activity and environment according to Schmidt et al. [226]. They consider this space to define a user's context. In their architecture, each implicit context (physical and logical sensors) is defined at a time t in the range of possible values D. On each sensor a set of cues each taking the sensor value up to a certain time t^i providing a symbolic or sub symbolic output in the domain of possible values E is defined. Thus, the values of one sensor may be represented by several cues. A context is derived from these cues. Hence, a context is described by a set of two-dimensional vectors h that each consist of a symbolic value v derived from the cues and certainty p as a probability in the reading, i.e. context = $\{(v_1, p_1),\dots$,

(v_n, p_n)}. The context is then utilised by scripting by the context-aware agent. The authors [226] recognised by their experiments three difficulties in such rule based context recognition: ambiguity, boundaries and the undefined context model, e.g. difficulties in recognising the context, operating close to thresholds and in undefined context(s).

In addition to means of representing context as variables or predicates, Zimmerman et al. [276] have quite a different way of representing context. The difference of this view with respect to the other described in this section is that Zimmerman et al. consider an "entity in the centre of a surrounding individual context" [276]. Hence, they consider what we call situations and how the user application migrates between them. Such a behavioural migration is interesting and certainly important in defining situations demanded by the user application to provide for a task; however, it is not related to how context is derived, to context dependency not to trustworthiness and is thus not considered further.

2.3.3 Context Acquisition and Binding

When an application acquires context from another application we call this binding. Hence, binding a context makes the context consumer dependent on the provider. Binding is either static or dynamic. In static binding, a context providing application is predefined. The model of static binding is simplified as it assumes a context to be available all the time at some quality. However, within a changing environment where qualities of a provider vary, dynamic context binding is motivated.

Broens [48] defined dynamic binding of a context as a 5-stage process: (i) discover context providers , (ii) select suitable providers, (iii) acquire the context by establishing a binding, (iv) monitor the context provider and eventually, (v) release the binding. They consider the discovery stage (i) with a kind of a broker that discovers context providers and the selection stage (ii) to rank applicable providers with the help of some user or of a predefined policy that determines their suitability. Of the suitable context providers, some are bound in stage (iii) meaning that the context of this provider is available until the binding is released. These contexts are monitored in stage (iv) in order to react to changes in their qualities and possibly initiate a new discovery phase. This continues until the releasing stage (v) as a consequence of termination or of a command to do so, e.g. in case the quality decreased below threshold.

Of this 5-stage process, we assume a context available as the system context $C_S(t)$, hence omitting the discovery process as mentioned in Section 1.5, limitation of scope. The selection stage (ii) is rudimentary defined by the context consuming agent, that for a user application is defined by $situation_A(t)$. Similar means could be applied for applications as well, in accordance to Sections 2.3.1 and later 3.3.2.3 where the context providing entity encapsulates its underlying contexts. The contexts eventually bound (as in stage (iii)) by a user application

may be considered as $situation_N(t)$ and S_N. These contexts are monitored (iv) before each round of execution. This is equal to releasing the bindings (v) after each execution.

The critical phase of selecting contexts to bind (ii) has largely been ignored [48] or is described at a high level. A binding decision is reasonably defined by a predicate. As a predicate is a policy, a drawback on the attempts to order available contexts is the policy's static nature. Henricksen et. al [121] do, however, note that user feedback may be used for adjusting the policy to better meet the user biases – further motivating the central role of the QoC parameter of trustworthiness.

2.4 Success criterion

QoC metadata parameters capture and represent the inherent imperfection of a context. QoC plays a significant role in the decision making. Hence, as context is derived, it is necessary to present realistic functions for deriving QoC parameters as well. This requires an accurate representation of context and QoC parameters giving rise to a success criterion:

Success criterion 1 A methodology in which structured context derivation including the QoC parameters is possible.

To meet Success criterion 1 in an open environment, the key lies in context abstraction. This abstraction implies that (i) context providers are to state QoC on the context(s) they provide and (ii) a means to compose the bound contexts' QoC parameters.

If Success criterion 1 is met, context may logically be reasoned about. This enables hiding the details of derivation from the user application logic declaring $situation_A(t)$ and consuming $situation_N(t)$. Moreover, the realisation of context consuming user applications dynamically binding at runtime appropriate context based on some QoC parameters becomes feasible. These aspects lead us to state Success criterion 2:

Success criterion 2 A rigorous means to model binding of a context in a user application.

If Success criterion 2 is met, a means for a best effort formal analysis on context is possible. Such an analysis would assume a context to remain unchanged throughout the actuator's execution.

"*We are drowning in information but starved for knowledge.*" John Naisbitt

3 Context Models and Context Derivation Architectures

In this chapter we build on the application types presented in Chapter 2 ***from the perspective of ubiquitous computing describing the context models, means of context derivation and the logical topology of derivation. We outline existing work on conceptual context models as well as existing logical topologies for context derivation, together with the used architecture for this. The success criterions of this chapter point out qualities of system architectures. Finally, we provide a contribution of this thesis in Section 3.5, motivating the logical context derivation topology to be a polytree in Section 3.5; this is a contribution of this thesis..***

All applications are engineered to provide or assist in providing a means for performing a task. A context-unaware agent operates algorithmically triggering actuation in response to a predefined sequence of instructions; whereas a context-aware agent adapts to the momentarily contexts [77]. Therefore, all agents that adapt the behaviour with respect to context are fundamentally context-aware [238] [276]. For this, the representation, means of modelling and derivation of context need to be outlined.

3.1 Introduction to Context Models and Context Derivation Architectures

Sensory devices capturing real world events are abstracted by applications providing elementary contexts. Deriving such low-level information to contextual information, both modelled as $context_S(t)$, is the key for the system to provide context requested by a user application, $situation_A(t)$. To derive on $C_S(t)$, the represented contexts relations need to be modelled. In this section, we will define and motivate issues regarding models for such structured derivation of context.

3.1.1 Context Modelling

There is no general context model capable of modelling all contexts. However, a context model is needed to define the context data [25] where a single model may provide for a family of context consuming agents. Consequently, in accordance to [83] we define a context model as follows:

Definition 11. Context model: "*the representation of contexts and their relations that may be relevant for a context-aware agent or a family of such agents.*"

This definition of a context model outlines a representation of an abstract view on relations of contexts. In this section we focus on the abstract model of relations of contexts; the representation of context information is considered in Section 2.3.2.

For the context model, we distinguish between two types of context models: the context acquisition model and the conceptual context model. The difference between these is that the conceptual model is independent of technological realisation. It is therefore primarily concerned with using contexts [25] [116]. Difficulties relate to modelling all conceptual contexts that the context consuming agent may require at the moment and in the future. Hence, a conceptual context model needs to distinguish between different contexts, while still be simple enough to provide a base for programming [259]. The conceptual context model is, therefore, often used and created at user application design time.

The context acquisition model describes a context's technical derivation, i.e. it is about detecting contexts [25]. This model is concerned with how a context and its QoC parameters are derived, i.e. about realising the conceptual model [83]; we discuss more on context acquisition models in Section 3.1.1.2.

3.1.1.1 Conceptual Context Models

The design of the conceptual context model should precede a detailed design of a context-aware application or user application [83]. Consequently, modelling contexts conceptually should be appropriate to [48] [85]:

(i) Characterise the context consuming agent's universe of discourse

(ii) Support common understanding, problem-solving, and communication among the various stakeholders involved in context-aware system development

(iii) Unambiguous representation of context

Statement (i) refers to the need to establish a realistic conceptual view that may be assumed by the context consuming agent [277]. Statement (ii) points out the need for a shared understanding of the contexts facilitating correct perception. This need was originally identified by Öztürk and Aamodt [277], e.g. *reachable*(*x*, *y*) is to be defined with *x* as a person and *y* as a device where *x* and *y* may have certain characteristics of their own. The last statement (iii) stresses the agreement on the representation of the context. This agreement has two dimensions, agreeing on conceptual representation and unit. For instance, *speed* may be relative to object (air, water, another car, earth's crust) and may be represented as m/s, km/h, mph. Noteworthy is that Broens [48] intends by unambiguity in (iii) merely unambiguity in perception, not unambiguity of the term. Hence, a conceptual context model defines the contexts that the context consuming agent may come to require.

Means to represent the context models are numerous and varied. Bettini et al. [34] is a survey on various existing models whereas Strang and Linnhoff-Popien [239] is a survey on classifying various context models for ubiquitous computing. Strang and Linnhoff-Popien [239] classify the approaches by scheme and data structure used to exchange contextual information to: key-value, mark-up scheme, graphical, object-oriented, logic-based and ontological modelling. Examples of these can be found in the survey [239]. They do identify shortcomings of the various models with respect to a classification. The survey [239] concludes that the ontological model is the most promising having, among others, a high degree of formality. This formality does, however, refer to the fact that ontology is machine readable [240]. The ontological context models do also suffer from being error prone, time consuming and having scalability issues when extending or modifying the ontology and they fail to capture imperfect contexts [127]. Moreover, the ontological models "fall short in offering their

users suitable sets of modelling concepts for constructing precise and explicitly characterized representations of their subject domains of interest" (*sic*) [116].

Hence, in this thesis we focus on logic-based context modelling. A logic-based conceptual context modelling has a high degree of formality but does not address partial validation, quality of information, applicability or incompleteness / ambiguity [239]. Except for applicability, this thesis addresses all the other shortcomings.

The pioneers of logic-based context modelling are Giunchiglia [105] and McCarthy [184]. Giunchiglia [105] views the context as a means to formalise the subset of an individual's knowledge used for reasoning. McCarthy [184] views context to specify the circumstance of a proposition or term. The fundamental difference between these views is that the former considers context as a partial view of the reality on which reasoning is applied, whilst the latter asserts for specifying more concretely the propositions and terms in a setting [104]; recall the context representation of Section 2.3.2. The logic-based context information modelling is considered further in Section 3.3.2.

3.1.1.2 Context Acquisition Models and Architectural Styles

Common to all conceptual context models is that they depend on some context acquisition model(s) for technological realisation. The context acquisition model is what Dockhorn Costa [83] calls the context information model and Perttunen et al. [205] the context representation model. It is a model on context derivation and establishes desired contexts from the history of observed contexts $C_S(t)$. The view is supported by the separation of concerns between context acquisition and usage [8] [25] [34] [56] [59] [60] [74] [76] [77] [83] [111] [165] [216] [227]. Moreover, a context acquisition model includes consideration of the QoC parameters.

The acquisition model is outlined by Chen et al. [59] to three general models: *direct sensor access*, *middleware* and *context server*. The direct sensor acquisition model refers to systems with locally embedded sensory devices, e.g. accelerometer, luminosity, acoustic sensors [103]. It has a logical topology of a one hop star with the user application in the centre. The architecture of direct sensor acquisition model constitutes the architecture of a first generation context-aware system [48] as depicted in Figure 7 inspired by [48].

The context middleware acquisition model is based on a layered architecture. The main tasks of a middleware is to transparently abstract the details of the underlying platform in order to facilitate reuse of the contextual information and perform functions that deal with common complexities [48]. With respect to the direct sensor acquisition model, the middleware model is extendable and provides transparency. The middleware acquisition model is depicted as the

second generation context-aware system in Figure 7 that performs elementary context discovery and selection.

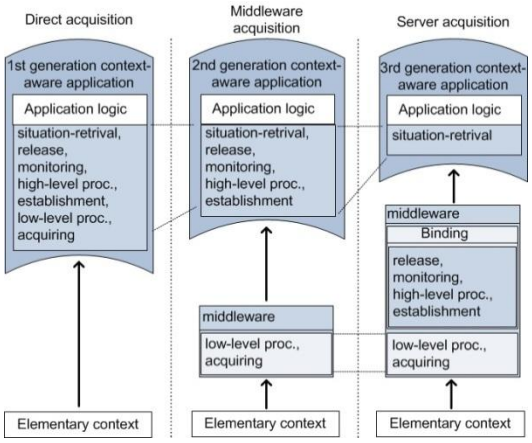

Figure 7: Context-aware system evolution

The context server acquisition model extends the middleware model by centralising contextual information in order to organise the information and permit multiple access to a context on a resource-rich device. As of transparency, the server masquerades the middleware. It is depicted the third generation context-aware system in Figure 7. In addition, a context server may derive complex contexts on behalf of the client, e.g. decide on binding, monitoring. Hence, a context server model is suitable when the contextual computations are overly resource intensive for the device running the user application. This is similar to that the user application would offload context derivation to the server [48], a view advocated by Hong et al. [132]. A concern with the context server acquisition model is, however, that of privacy. That is, a server may need a user's profile for deriving context, as is the case for the interactive system framework by Hong et al. [134].

An example of a context server acquisition model is the event centric Context Toolkit architecture [74] [77]. The Context Toolkit framework abstracts resources by *interpreters*, *aggregators*, *services*, *widgets* and central *discoverers*. In the Context Toolkit, a context discoverer is a context consuming agent's initial point of contact that maintains a registry over the framework's resources. A context consuming agent may subscribe the discoverer for notifications of changes in the resources and use it to locate the resources. Discoverers may form a hierarchy [74], e.g. a root discoverer with sub-discoverers. The context derivation itself is abstracted by widgets. Widgets may be subscribed to by other widgets, aggregators or a context consuming agent once discovered. Dey et al. consider the widgets to "abstract context information to suit the expected needs" [77].. Hence, widgets may form a hierarchy in their own right. An aggregator aggregates context providing a simplified operation for the context consumer

through which to acquire contexts. They also note that an aggregator has similar capabilities to a widget, but do not mention explicitly whether or not aggregators may form a hierarchy. We assume this as we are unable to find reasons objecting a hierarchy of aggregators. An interpreter in the framework may be used by a widget or a context consuming agent as a 'procedure', e.g. transforming location coordinates to an address. These are motivated by reuse. Services of the framework are what we call actuators that may be triggered by a widget, i.e. active context-awareness.

Other examples of implementations of context server acquisition models include TEA by Schmidt et al. [226] and the management framework by Filho et al. [96]. In Section 2.3.2 we have considered TEA. Filho et al. [96] implement a context server model with the server called a *context information service* that provides views for a user application in response to queries. Whether or not this service is specific to a set of users remains unclear. The model propagates sensors readings much alike our division. Their model reasoning does, however, rely on a (unspecified) context management administrator to define rules and QoC thresholds.

This separation of concerns and openness of the context acquisition supports architectures where the conceptual models technical realisation depends on an independent set of providing applications [243]. For more extensive overviews on context acquisition models, we direct the reader elsewhere [25] [39].

3.1.2 Logical Topology of Context Derivation

The logical topology for deriving a situation from elementary context is in the simplest case a one-to-one mapping [88]. This is the case for direct sensor access acquisition [59] [103]. In this case, the elementary context is directly consumed by the user application. More sophisticated system architectures, such as the middleware of a logical mesh topology, include hierarchically organised applications, abstracting the actual sensing and lower level context-derivation from its context consumer. This logical mesh topology should support central contexts, e.g. location, to be used by several higher level contexts, called a context colony [66] and context sources and managers hierarchy pattern [83].

From a context consuming agent's point of view, the contexts manifesting the information space have been represented as a directed (oriented) graph [66] [166] and as a directed acyclic graph (DAG) [35] [45] [83] [195]. The directed graph can per definition be cyclic. However, in this thesis we consider context depending indefinitely on itself infeasible. The DAG, on the other hand, is acyclic.

Modelling the logical topology of deriving a context $x(t) \in situation_A(t)$ as a DAG $G = (V, A)$ with a set of vertexes $V = \{v_i\}$, $i = 1, \ldots, n$, and directed arcs $A = \{(v_j, v_k)\}$, $j \neq k$, with direction from v_j to v_k may be interpreted as that v_j is the context consuming agent depending on context provider v_k. For an illustration,

consider Figure 4context derivation where dependencies are the opposite direction to information flow. Here, each vertex represents an application and given $(v_j, v_k) \in A$, v_j consumes (depends on) the data that v_k provides. Because G is acyclic and finite, there is at least one vertex with an *indegree* $\deg^-(v) = 0$. In terms of context derivation, this vertex necessarily provides a context $x(t) \in situation_A(t)$ or is void. Intermediate vertices with $\deg^-(v) \geq 1$ and outdegree $\deg^+(v) \geq 1$ all provide and consume contextual information, i.e. they are applications.

Obviously, some vertex providing context $x(t) \in context_S(t)$ may provide a situation $x(t) \in situation_A(t)$ simultaneously to have an indegree $\deg^-(v) \geq 1$, i.e. a context is part of a user application's situation simultaneously to a context for some other contextual information. An exception are the elementary contexts that always have an outdegree $\deg^+(v) = 0$, i.e. they do not depend on any other context. The abstraction of context in a graph G means that any vertex needs only to be concerned with vertices that it refers to, i.e. given $(v_j, v_k) \in A$ where $j \neq k$, v_j is concerned with v_k whereas v_k is not concerned with v_j. Hence, a provider abstracts a set of vertices from its context consumer. This model follows the architectural style with a principle of limited visibility [243].

Regardless of the logical topology, each vertex needs a mechanism for incorporating a support for trust, security and privacy as well as history management and discovery / recovery [65] [214]. Of these, source discovery / recovery are out of the scope of this thesis. Trust, security and privacy are to protect the subject from revealing unwanted information where security and privacy are matters of policies on the provider side, e.g. level of encryption and abstraction. Whether Coutaz et al. [65] or Rey and Coutaz [214] consider trust as a policy for access control or as experience-based is not stated clearly. In addition, they do not outline the use of history in greater detail, i.e. whether history refers to $C_S(t)$ and if it logs behaviour remains unclear. Hence, determining whether trust refers to policy-based trust or experience-based trust is not possible; this matter is considered in detail in Chapter 4.

3.1.3 Deriving with QoC Parameters

Each context is accompanied by the QoC metadata capturing the inherent imperfection of contexts. Hence, an application consuming context and providing another context needs to calculate on the QoC parameters as well. This is motivated by research relating to indirect acquisition models [35] [66] [78] that report QoC an important factor to consider. Moreover, research addressing the QoC as a concept in its own right typically lists a set of parameters and motivates their importance [54] [161] [237].

Manzoor et al. [176] claim that their work is the first that "presents and evaluates the QoC parameters as the worth of context information for an application and provides the context information enriched with these QoC

parameters" [176]. They provide equations on deriving up-to-dateness, trustworthiness, completeness, and significance. However, we note that their view on QoC parameters is very different from ours with the exception of up-to-dateness. They relate trustworthiness of an object to the distance between the sensor and the evaluated / measured subject. The latter has to be within a defined threshold that, when multiplied with the accuracy of the sensor provides the level of trustworthiness. In their motivating case the sensor is a camera and the subject the photographed object. Thereby, what Manzoor et al. [176] call trustworthiness is a mixture of QoC parameters of precision and resolution rather than a measure derived from experiences. Moreover, their completeness as a measure of "quantity of information that is provided by a context object" or significance as "indicates the worth or the preciousness of context information" [176] does not fully match any QoC parameter outlined by Buchholz et al. [54]. Of these, the significance parameter relies on the context provider's user profiled value, i.e. as if each sensor had a specific purpose and used within a specific type of user applications only.

Filho et al. [96] present an interesting approach to derive on QoC parameters considering sensitiveness, access security, completeness, precision and resolution on context. They map these to a relative value in [0, 1] by relating to the number of measureable parameters. Recall the QoC parameters presented in Section 2.1.6; as the algorithms of Filho et al. [96] might suffice for the factual parameters precision, granularity and freshness, they conclude that providing a function on probability of correctness or trustworthiness is part of future work. We provide our view on the parameter of trustworthiness in Chapter 4.

3.2 Problem analysis

A context-aware architecture is separated by concern to context derivation populated by applications providing $context_S(t)$ and context utilisation populated by user applications declaring $situation_A(t)$. On $situation_A(t)$, the net situation $situation_N(t)$ defines the relevant contexts of the user application logic at time t. Hence, the enabled context-aware functionality of a user application is determined by $situation_N(t)$ and $S_N(t)$ specifying how, if at all, to adjust. This leads us to state Challenge 3.

Challenge 3 Defining an architecture that abstracts details of context derivation from its context consumer without loss of QoC information.

Challenge 3 is paramount as it constitutes the key for open, decentralised control and derivation of a context that a context-aware user application depends on.

As the context derivation is done in an acyclic manner by autonomous applications, the context consumer should adapt to changes in the context or its qualities. This leads us to stating Challenge 4.

Challenge 4 Providing a methodology enabling dynamic runtime binding of context providing applications.

Challenge 4 is critical for providing efficient and prompt contexts. It requires great flexibility of the methodology it is expressed in. Together, Challenge 3 and Challenge 4 yearn for an open architecture populated by autonomous components that derive high quality $context_S(t)$.

3.3 State of the art

Defining the possible contexts ($C_S(t)$) serving a user application its desired situations ($S_A(t)$) at design time is very difficult or impossible. This is because context is a dynamic construct and defining a priori all relevant contexts is not feasible [112]. The dynamicity relates, among others, to informal updates, e.g. mobility and changes in quality. As many of these informal updates may be captured and defined in the conceptual context model, the modelled contexts' changing qualities may not as it derives from elementary contexts. Such changing quality of contexts has only been sparsely considered with Manzoor et al. [176] claiming to be the first to address the value of the context. Perhaps this is because the focus of context acquisition models has been on supporting ubiquitous applications rather than on modelling the context [83]; a trend already noted by Gwidzka in 2000 [119] and later supported by Soylu et al. [238].

Assuming separation of concern between a context provider and the context consumer, the context provider or its qualities are not affected by its consumer. However, as the context consumer critically depends on the provided contexts, it may in addition to defining policies as predicates on the consumed contexts also strive to bind as high-quality context providers as possible. As a consequence, the context provider's underlying derivation architecture should propagate elementary context's imperfection for the context consumer.

In order to provide QoC parameters, this section presents state of the art logical topologies of context derivation, logic based context acquisition models and conceptual system architectures.

3.3.1 The Logical Topologies for Context Derivation

As noted in Section 3.1.2, the logical topology for deriving contexts is quite frequently considered a DAG [35] [45] [83] [195]. Consider a DAG $G = (V, A)$ with vertices $V = \{v_i\}$, $i = 1, \ldots, n$ and directed arcs $A = \{(v_j, v_k)\}$ where $j \neq k$ and $v_j, v_k \in V$. This type of graph allows undirected cycles, e.g. $\{(v_x, v_y), (v_y, v_z), (v_x, v_z)\} \subseteq A$ where $x \neq y \neq z$. Such undirected cycles are unacceptable in context derivation of imperfect contexts in a middleware acquisition method; these, similarly as in Broens [48] and in Section 3.1.1.2 are considered to abstract the

details from its context consumer for the sake of reuse. That is, if $\{(v_y, v_{x1}), (v_z, v_y), (v_z, v_{x1}), (v_z, v_{x2})\} \subseteq A$ then v_z may not know that the context it acquires base only on two elementary contexts, v_{x1} and v_{x2} and not three. This is troublesome whenever the amount of disjoint readings affect the outcome, e.g. in case of average in addition to calculations on trustworthiness as presented in Chapter 4. Hence, a DAG is not sufficiently restrictive and a logical topology prohibiting undirected cycles is needed. To the best of our knowledge, we are the first to acknowledge this shortcoming in terms of context derivation. We present a novel view on this problem defining the logical topology to be a polytree in Section 3.5.

3.3.2 Existing Context Models

Most existing approaches of context acquisition models focus on context discovery and communication rather than on modelling the context [83]. Moreover, early context models where typically chiselled for providing for a specific task or family of tasks [239]. These statements are supported by Soylu et al. [238] stating that "approaches presented in current literature…" do "…not really manage to go beyond the borders of traditional computing" [238]. Perhaps this border is in including the user's intents and biases, i.e. what Gwidzka [119] called the internal context. In addition, with respect to the definition of a context model in Section 3.1.1, the reasoning in favour of a logic-based approach in Section 3.1.1.1 and the logical topology of a DAG, this section considers conceptual models that adhere to these requirements.

Hence, in the following we consider logic-based conceptual context models. We outline Context Modelling Language (CML) in Section 3.3.2.1, the situation lattice in Section 3.3.2.2 and other relevant logic-based context models in Section 3.3.2.3. The other models comprise of Loke's abstract model, the situation lattice model and Dockhorn Costa's graphical notation of the situations. For a more comprehensive review on context models we direct the reader to Bettini et al. [34] and Strang and Linnhoff-Popien [239] whereas for a review on means to identify a situation from contexts and their models' enabling technologies to Ye et al. [270]. In Ye et al. [270] logic-based models are called specification-based context identification.

3.3.2.1 Context Modelling Language

The CML is a graphical, still formal object role modelling language by Henricksen et al. [120] [126]. It models roles between concepts within fact types (contexts) assuming a closed world, i.e. known in detail. This model provides a means for reasoning on the fact types by abstract high-level context defined in predicate logic [34] [121] called situation predicates, $S(v):\varphi$ where S is the name of the high-level context, v a set of variables and φ a well-formed logical

expression over free variables *v*. The logical expression employs comparison operators (≤, ≥, =, ≠…), logical connectives (∧, ∨, ¬,…) and arithmetic operators (+, -, ×, ÷,…). It is defined on a finite space of immediately bound variables {x_1, …, x_m} constrained by an assertion $r[y_1, \ldots, y_n]$ where $r \in R$ of relations on the model *I* and $I(r)$ denotes the set of tuples in *I* that belong to a relation $r \in R$. Then an assertion $r[y_1, \ldots, y_n]$ is *true* in *I* if $<y_1, \ldots, y_n>$ is a tuple in $I(r)$ and *false* otherwise. That is, given that $r[y_1, \ldots, y_n]$ evaluates true, $<y_1, \ldots, y_n>$ restrict the possible values for the set of bound variables {x_1, …, x_m} as {x_1, …, x_m} ⊆ {y_1, …, y_n}.

Hence, in line with Henricksen and Indulska [121], the predicate is of the form:

$$\square x_1, \ldots, x_m \bullet r[y_1, \ldots, y_n] \bullet \varphi$$

Here □ is a placeholder for either ∀ or ∃ and ● is a mere separator lacking semantics. Informally, this predicate may be read as '*there exists at least one variable* $x_1, \ldots, x_m$ *that satisfy* $r[y_1, \ldots, y_n]$ *for which formula* φ *holds*'. Moreover, combining abstract high-level contexts predicates, say $S(v_1):\varphi$ and $S(v_1):\psi$ to a composite *C* predicate of $CS(v_1, v_2):\varphi \wedge \psi$ is straight forward. Hence, $CS(v_1, v_2):\varphi \wedge \psi$ evaluates identically to the conjunction of its atoms $S(v_1):\varphi \wedge S(v_1):\psi$ closed under ¬ , ∧ and ∨ [120]. Obviously, such combinations of context predicates support reuse and are considered in terms of this thesis as $context_S(t) \in C_S(t)$.

Given a context model with temporal fact type *engagedIn* over person, activity, start time and end time, an example of a situation predicate may be:

occupied(*p*): $\exists t_1, t_2, activity \bullet engagedIn[p, activity, t_1, t_2] \bullet ((t_1 \leq t_{now} \wedge (t_{now} \leq t_2 \vee t_2 = null)) \vee ((t_1 \leq t_{now} \vee t_1 = null) \wedge t_{now} \leq t_2)) \wedge (activity = \text{'in meeting'} \vee activity = \text{'taking a call'})$

Here *p* is the bound variable denoting a person and start time t_1, end time t_2 and *activity* are free variables. Examples of a composite predicate and a predicate involving a probability are:

isReachable(*p, c*): $\forall d \bullet requiresDevice[c, d] \bullet locatedNear[p, d] \wedge permittedToUse(p, d)$

locatedAt(*p, pl*): $\exists prob \bullet personLocatedAt[p, pl, prob] \bullet prob > 0.8$

Here *p* stands for person, *c* for channel (means), *d* for *device*, *pl* for place and *prob* for probability. Informally, *isReachable* defines the composite situation of a person *p* being reachable on channel *c* so that all the devices required to use *c* are located near *p* and *p* is permitted to use these devices *d*. CML also provides quality annotation in terms of certainty as a probability on a fact, e.g., situation predicate *locatedAt* is referred to with a probability and a threshold. More examples are found in referenced work [120] [121] [126].

CML does not address uncertainty; however, Henricksen and Indulska [121] [122] extend their model to address 'unknowns' and 'ambiguity'. These are defined on the assertion that the tuple $<y_1, \ldots, y_n>$ is not a tuple in $I(r)$ but may become one by replacing one or more y_i by *null* or when $r[c_1, \ldots, c_n]$ is

ambiguous as was the case for *locatedAt*. Hence, the assertion defines what is demanded and by relaxing the demands to *null*, the result is unknown. Having this third value of 'unknown', an assertion is *false* whenever it is neither unknown nor *true* [121] [122], i.e. when it is certainly *false*.

The weakness of CML is that it assumes a closed world [121] [122] making it suitable for conceptual modelling with a complete view on the domain it models but not for evolving context acquisition models. In addition, CML is a flat model with respect to the means of context derivation, i.e. all contexts are represented as atomic facts. Hence, if having some dominant context or a hierarchical structure, other models may be more appropriate [34]. A consequence of this is that CML is well suited for development of models for specific applications or application domains [34]. These weaknesses have given rise to hybrid models combining the benefits of graphical models and ontological modelling [127].

3.3.2.2 Situation Lattices

Considering contexts as terms of predicates in a lattice structure was initially proposed by Woods [261]. Later, the related formal concept analysis of context as a concept lattice [258] was proposed. The concept lattice may be used to define the hierarchy of concepts with respect to their common attributes from some given relation [49]. It is noted to suite backward chaining whenever a context under inspection is chosen beforehand, as well as for forward chaining, when deriving elementary context to context(s) by inference rules [268] [269].

A situation lattice is a lattice $L = (C, \leq)$ where C is a set of non-exhaustive contexts ordered by the partial order $\leq$ specialisation relation. Hence, the lattice is a dependence structure when viewing it bottom up, and conversely, a generalisation viewing it top down. For contexts $c(t)$, $d(t) \in C$, if $c(t) \leq d(t)$ then in terms of context we say that $c(t)$ is a more abstract level of context than $d(t)$, i.e. that the conditions for $c(t)$ to hold are stronger than those of $d(t)$. This assumes $c(t)$ and $d(t)$ to be defined on the same dimensions [238], i.e. have the same accepted values. Let a predicate p_c characterise values when $c(t)$ evaluates true and p_d characterise values of $d(t)$, then if $c(t) \leq d(t)$ it holds that $p_c \Rightarrow p_d$ [268] [269]. E.g. with respect to Figure 8, as *grp_meeting* $\leq$ *talk* and *talk* $\leq$ *speaker* this means that $p_{grp\_meeting} \Rightarrow p_{talk}$ and $p_{talk} \Rightarrow p_{speaker}$ that by transitivity $p_{grp\_meeting} \Rightarrow p_{speaker}$. Hence, the predicates are partially ordered by the implication relation ($\Rightarrow$) defining abstraction / generalisation. Of any set of elementary contexts, their meet is their most general situation. Hence, the weakest predicate of all is *true*, element $\top$, that holds true for any configuration, i.e. whenever the system is running properly; and contrary $\perp$ identified by predicate *false* being the strictest possible condition that is *true* for no configuration at all, i.e. for the improper context [269].

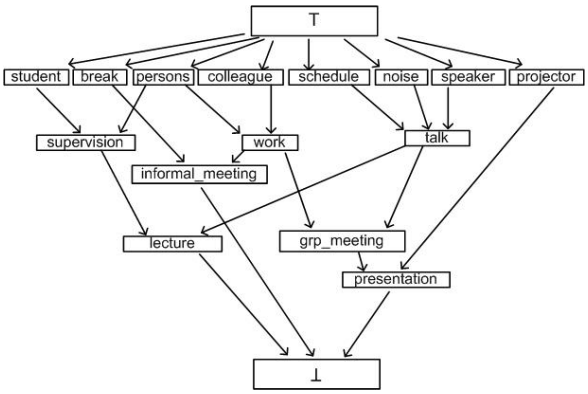

Figure 8: A situation lattice for meetings

Figure 8 depicts a lattice model of context for a model of the domain '*types of academic meetings*'. Considering Figure 8, for example context *persons* as number of persons as well as *work* adheres to that a context should be available at different phases of processing [83] and have the ability to contribute to many (disjoint) higher-level contexts [25] whose meet is ⊥. Consequently, this view concurs with the notation of CML $CS(v_1, v_2):\varphi \wedge \psi$ where φ and ψ are predicates in their own right over propositions combined by logical connectives with *join* ⊤ and *meet* ⊥. A proposition for context *work* may for some model be defined as $persons \geq 2 \wedge \forall persons: persons \subseteq colleague$. Moreover, Figure 8 addresses that of encapsulating underlying contexts from the view of a more general context [238]. That is, assuming shared dimension and values, *group_meeting* is not directly concerned with the predicate of *work* but with its encapsulating predicate of *group* providing more detail.

The main drawback of modelling context derivation in a situation lattice is the irreversible abstraction by a predicate [268] [269]. When so, the QoC parameters are abstracted as well or some novel means of propagating these need to be found.

3.3.2.3 Other Context Models

A model that is related to the situation lattice focusing on situation recognition is proposed by Loke [173]. This formal model considers a white-box context-aware system as (Σ, Π, Θ) where Σ denotes the sensors, Π denotes the interpretation and Θ the situation reasoner. The finite sensors $\Sigma = \{\sigma_i\}$ for $i = 0, \ldots, n$ produces with time t a set of (history) sensor readings $G_i \in \boldsymbol{G}$. The interpreter Π performs a mapping from $\boldsymbol{G}$ to a context $C \in \boldsymbol{C}$, e.g. noise provided as a Boolean with respect to a threshold instead of decibel or *persons* as number of persons. They consider C to be grounded in some ontology, i.e. consist of the elementary contexts with respect to the situation lattice. The interpreter $\Pi \subseteq (\boldsymbol{G} \times \boldsymbol{C})$ applies each G_i to a set of contexts $\{C_1, \ldots, C_n\}$. Moreover, the situation

reasoner Θ defined as a pair of relations (Θ_C, Θ_S) maps in Θ_C recognised contexts $\mathcal{P}(\boldsymbol{C})$ to situations and in Θ_S recognised situations $\mathcal{P}(\boldsymbol{S})$ to higher level situations. More precisely, when $\boldsymbol{S}$ denotes the set of situations we have that:

$\Theta_C \subseteq (\mathcal{P}(\boldsymbol{C}) \times \boldsymbol{S})$
$\Theta_S \subseteq (\mathcal{P}(\boldsymbol{S}) \times \boldsymbol{S})$

Loke [173] notes that these situation reasoners take context and derive situations or aggregates situations to derive more situations. Let the set of contexts $\boldsymbol{C}$ be {*student*, *break*, *persons*, *scheduled*, *noise*, *colleague*, *speaker*, *projector*} and the set of situations $\boldsymbol{S}$ be {*work*, *talk*, *supervision*, *informal_meeting*, *grp_meeting*, *lecture*, *presentation*} as in Figure 8. A few relations in Θ_C and Θ_S are:

({*scheduled*, *noise*, *speaker*}, {*talk*}) $\in \Theta_C$
({*persons*, *colleague*}, {*work*}) $\in \Theta_C$
({*talk*, *work*}, {*grp_meeting*}) $\in \Theta_S$

This forms an incremental approach in building the situations and contexts. Any change in any context or relation may change the situation [65].

Loke [173] further defines the actuator A on the recognised contexts and, what they call, the recognition power of the system as module M that maps recognised contexts by recognised change to an action A, i.e. M: $\mathcal{P}(\mathbf{C})$ × 'recognition change' → A. Hence, a context-aware system with actions is defined ((Σ, Π, Θ), M), e.g. let x be a person and y a room, then if ({*x not in y at* t_1, *x in y at* t_2}, {*x enter y at* t_2}) $\in \Theta_C$, then (({*dark in y at* t_1}, {*x enter y at* t_2}), *turn lights on y*) $\in M$. For more detailed description, the reader is directed elsewhere [172] [173].

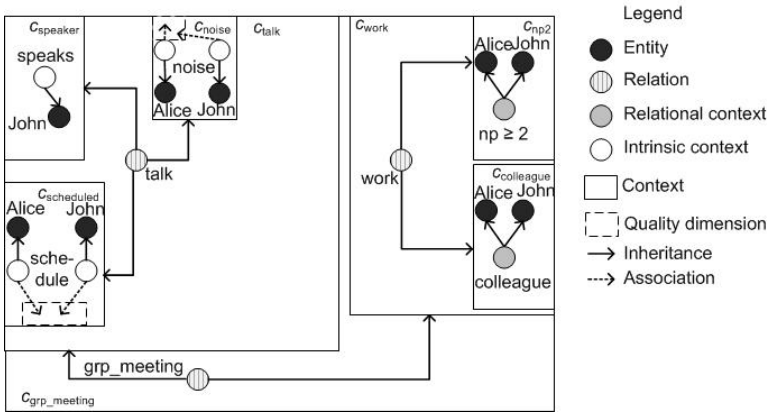

Figure 9: An example of situation of contexts

In addition to Loke's [173] model, Dockhorn Costa et al. [85] propose a graphical context model representation as contexts (they call our context a situation) that are genuine ontological entities. In their model, a context may

generalise another by encapsulating it [85]. They distinguish between *intrinsic* and *relational* context with an intrinsic context describing a single entity whereas relational context describes the relation among several entities always manifesting a context. Moreover, they allow a context to depend on an intrinsic context, e.g. connection context is characterised by start and end time. The model also visualise that any change in any context or mapping relation may change the general context depending on it [65]. Obviously, several contexts may hold simultaneously. In Figure 9 we illustrate a sublattice of the specific context *grp_meeting* $L_{sub} = (C', \leq)$ of Figure 8 to which the implicit relations of a considered ontology are added explicitly for readability.

3.3.3 A Context Derivation Architecture

We consider a context-aware ubiquitous system consisting of three complementary subsystems: (i) a system capturing elementary contexts by mapping real world event to software events, (ii) a system reasoning on acquired contexts (software events) and (iii) a system mapping software event(s) to real world events for providing a user means to perform a context-aware task. It follows the typical model outlined by Loke [173] and supports separation of concern that is considered fundamental in numerous related works [8] [25] [34] [56] [59] [60] [74] [76] [77] [83] [111] [165] [216] [227].

Recall Figure 4 for an outline of an architecture considered in this thesis supporting the constraints mentioned above. In that architecture, the applications providing elementary contexts capture the real world events (i); applications deriving on elementary contexts provide contextual information (ii) and the user application based on its logics may or may not trigger an actuation (iii). This architecture is inspired by several frameworks [31] [94] [95] [111] [121] [124] [133] [166] [173] [238] [270].

3.4 Success Criterion

In ubiquitous computing, the middleware populated by autonomous applications deriving the contexts is transparent to the user application. This is necessary for providing scalability, reuse and efficient utilisation of the available imperfect contexts motivating the separation of concerns. It does, however, stress the importance of representation of context in a model, specification of context logically and reasoning on the propagation of imperfect contexts; noted “the principal research topics on situation identification” by Ye et al. [270]. This involves considering QoC parameters.

Reasoning on the QoC parameters leads us to stating Success criterion 3:

Success criterion 3	Defining a dynamic, scalable, transparent hierarchical architecture for providing context accompanied by QoC parameters.

Assuming rigorous representation of context, Success criterion 3 stresses the importance of reasoning with QoC as a part of the architecture.

If Success criterion 3 is met, the context derivation architecture may provide for a truly transparent framework in which situations are reasoned about. The limitations relate more to the expressivity of the QoC parameters that in this case provide means for expressing biases through the parameter of trustworthiness. Hence, realising Success criterion 3 is a key in shifting the information age to the information revolution where a ubiquitous computing paradigm would truly be omnipresent to its user. Such a shift would provide a means to customise / personalise adaption where each user application may utilise only situations of its preferences.

3.5 An Undirected Acyclic Context Derivation Topology

A DAG modelling context derivation is insufficient, as noted in Section 3.3.1. The acknowledged reason is the undirected cyclicity of a DAG. Hence, modelling a context to be derived in an undirected acyclic graph is necessary. On such a graph, the dependency relation may induce direction for each arc. This problem setting is indirectly noted by Dockhorn Costa who state that "a system component can provide a service, but at the same time it can shield a whole composition of services from its service users" [83]. If the provider shields the contexts it depends on from its context consumers, this implies that only undirected acyclic graphs qualify for providing contexts consumed by an application.

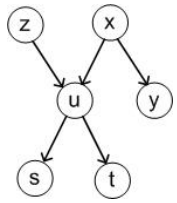

Figure 10: A polytree

A DAG with at most one undirected path between any two vertices ensures shielding of a composition and undirected acyclicity. With undirected paths we refer to making all arcs undirected e.g. in terms of a DAG with $V = \{u, x, z\}$ and $\{(x, u), (z, u)\} \subset A$ there is an undirected path between x and z. Such a graph is per definition a polytree; Figure 10 depicts the following polytree $V = \{s, t, u, x,$

$y, z\}$ and $A = \{(x, u), (z, u), (u, s), (u, t), (x, y)\}$. Each polytree is a multitree, i.e. a DAG where a subgraph reachable from a node forms a polytree in its own right. The undirected acyclicity of a context $x(t) \in situation_A(t)$ implies that any context may contribute with at most one view per $x(t)$ and no two contexts $x_i(t)$, $x_j(t)$ may share more than one subgraph. Hence, a context may only be used to derive higher level contexts with disjoint set of dependencies. That is, a single vertex $w \in W$ may have an outdegree $\deg^+(w) \geq 2$, but the intersection of the reachable nodes needs to be $\emptyset$, e.g. for the depicted polytree V depicted in Figure 10 $\deg^+(x) = 2$ but $\{y\} \cap \{u, s, t\} = \emptyset$.

Moreover, all vertices a vertex "shields" have a compatibility relation [104], establishing a context providing a more comprehensive view of the environment [105], e.g. vertices u and y have a compatibility relation with respect to x. Thereby, each context $c_i(t)$ is a view of a set of contexts it depends on (are reachable) at a given time t. Consequently, a polytree appears as a valid logical topology for context derivation.

"*There are two important entities on web: people and information*." – Jennifer Golbeck 2009 [107]

4 Trustworthiness as a Parameter of QoC

In this chapter we consider trust-aware context. We outline properties of a trust relation as well as types of trust, motivating the need for the QoC parameter of trustworthiness to be experience-based. On the experience-based trust, the levels of trust, a generic model for representing trust as well as the networks of trust relations are presented. For representation, we outline the difference between trust as a variant of Dempster-Shafer theory and trust modelled by probabilistic systems. On these, in the state of the art section we consider computational models with an emphasis on Subjective Logic. Finally, the success criterions are presented.

The ability to trust is fundamental for the existence of the human society [87]. It is the mental state that enables collaboration, formation of groups, feelings of relative security etc. [58]. Moreover, trust enables a feeling of reliance in inherently inaccurate and imperfect matters, e.g. how trustworthy is a context and finally, may the context be trusted. This 'feeling' is something that only cognitive entities having internal explicit goals (intents), hereafter the trustor, may perceive in some other uniquely identifiable matter, hereafter the trustee [58] [93]. Hence, in terms of this thesis, the QoC parameter of trustworthiness depicts the extent to which a trustor (the consuming entity) relies on the trustee providing a context accompanied by QoC parameters; for the terminology, (un)trustworthiness refers to a level of trust whereas (un)trusted is a Boolean level; trust is used as a general term for these.

Trust in computer science is considered either *policy-based* or *reputation-based* [40]. The policy-based trust, also called resource access trust [110], was originally introduced by Blaze et al. [37] as a variant for specifying security policies of a resource in terms of credentials and relationships for authorisation. Implementations of policy-based trust include access control, firewall rules, logical constraints. Common to all of these is that the level of trustworthiness is decided by a policy, i.e. by a predefined Boolean rule making the trustee (un)trusted with respect to the proposition.

The reputation-based trust is sometimes used interchangeably with the term *experience-based* trust that we prefer hereafter. This describes a level of trustworthiness based on a priori recorded experiences. In addition to the first hand experiences a trustor possess in a trustee, the level of trustworthiness may be ascertained by experiences acquired from referral entities, i.e. the reputation. As each experience level of satisfaction is evaluated by the trustor, the experience-based trust becomes similar to the human notion of trust, i.e. it is dynamic, emergent, incomplete, relative and subjective. Computational models on such a human notion of trustworthiness "aims at supporting a decision making by computational agents in the presence of unknown, uncontrollable and possibly harmful entities and in contexts where the lack of reliable information makes classical techniques useless" [163]. Commercial implementation areas of experience-based trust include online auctions, product review sites and discussion forums, to mention a few.

Because of these characteristics, policy-based trust and experience-based trust are nearly reverse views of each other: in experience-based trust the resource consumer (trustor) evaluates the provider (trustee) whereas in policy-based trust the resource provider (trustor) evaluates the consumer (trustee) [150]. As in this thesis we consider the QoC parameter of trustworthiness [54], we refer to the level of trustworthiness a consumer (trustor) perceives in the provider (trustee), i.e. experience-based trust. However, policy-based trust may be relevant to certain context-aware settings. As a consequence, hybrid trust models implementing both experience-based and policy-based trust have also been introduced in the literature [57] [162]. Notable is also that when an experience-

based trust level is used in a Boolean decision, a policy is applied on it; this defines the experience-based trustworthiness levels as supportive parameters.

With experience-based trustworthiness excluding the classical techniques, i.e. probabilistic systems, this thesis studies an alternative, the Subjective Logic that is based on Dempster-Shafer theory. Hence, we consider a Bayesian probability from the subjectivist view measuring a 'personal belief' rather than objectivist view treating probabilities as an extension of logic. That is, we do consider the probability of provability as opposed to the probability of truth, i.e. we consider trustworthiness probabilities as a representation of the natural language words 'belief', 'doubt', 'evidence' and 'support' [204]. From this level of trustworthiness featuring a level of uncertainty we outline a means to compute the context based on weighted contextual average on a deterministic domain with compatibility relations.

4.1 Trust and Trustworthiness

The experience-based trust is responsible for overloading the concept of trust [162]. It is typically defined in accordance to Gambetta [102] stating that: "*Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends*" [102]. This definition is called 'reliability trust' by Jøsang et al. [150]. However, as we do not seek for a means to merely model trust, but use it as a parameter of context supporting a context-aware decision, trust outlines a level of relative security, called trustworthiness. Here relative security refers to the free will to jeopardise welfare, hence negative consequences are possible. Therefore, in this thesis we define trust based on the broader definition of McKnight and Chervaney [187], called 'decision trust' [150], that we consider to include Gambetta's [102] 'reliability trust'. However, we include that the trustee does not need to be a party (that refers to an agent or group of agents) but may be a matter of any kind [58], e.g. a car. Moreover, with respect to the terms as used in this thesis, we note that this definition defines trustworthiness:

Definition 12. Trustworthiness: "*The extent to which a trustor is willing to depend on a trustee in a given situation with a feeling of relative security, even though negative consequences are possible.*"

This definition, even though general, includes two fundamental assumptions. First, we observe that trustworthiness is relevant only when something can go wrong. Hence, the concept of trustworthiness is a 'feeling' of unwarranted expectations that a trustor perceives in a trustee and trusting something certain is void. Secondly, trustworthiness is situation dependent. That is, trustworthiness captures the subjective probability that the trustee will conform to the intents of the trustor in a setting at a moment of time. The claimed behaviour of the trustee

is captured by the QoC parameters whereas the setting is called the *proposition*. On a proposition a level of trustworthiness is expressed, e.g. in the event of *picking a ball from a bowl* the level of trustworthiness is in the proposition *ball is green*.

Hence, all the transactions where expressing trust is valid involve some risk as well [58], where risk denotes the realisation of the negative consequences of the definition, e.g. if the picked ball is *red*. Obviously, should this risk realise, the level of trustworthiness on the proposition is to be decreased [131] [177] and conversely, if the trustee provides as expected, the level of trustworthiness should increase. The relation between risk and possibility for the trustee to conforming to expectations multiplied by the importance of the event at hand is what Marsh and Briggs [178] call cooperation threshold. This cooperation threshold is fundamental for decision support that, as it turns out, may motivate engaging in a transaction with a less a trustworthy provider when in great need of the offered service. To calculate this relation, a utility function has also been defined [145].

4.1.1 Properties of a Trust(worthiness) Relation

The single most important aspect of a trust relation is the unique identification of the entities. Assuming this, trust, and symmetrically not to trust, describes a level of reliance a trustor perceives on a trustee. On such a relation, there is a wide agreement on central properties [265]. Below we list some of them including a motivation as to why this is the case. We omit references and note that foundational research, such as Grandison et al. [110] agrees with these.

Trust property 1: Trust is *subjective*

As of the subjectivity, a level of (un)trusted or (un)trustworthiness perceived in a trustee may vary between trustors due to the non-uniformity of available experiences and/or appreciation. Hence, trust on a trustee is a specific trustor's perception. This motivates a non-universal level of trust, i.e. entity *A*'s and entity *B*'s perceived level of trust in a matter *C* may differ.

Trust property 2: A trust relation is *asymmetric*

Simply, if *A* trusts *B* in proposition *x* to a level *y*, then nothing about *B*'s trust on *A* in *x* may be derived from this. Hence, a trust relation is always directed motivating asymmetry.

Trust property 3: Trust is *incomplete*

Here, incomplete is used as a substitute for not dogmatic, i.e. trust is non-additive. This is the case for all informal acts [131]. If trust was dogmatic, it would be void and the relation treatable by objective probabilistic logics. The motivation is that not even a trustor may trust itself completely. Hence, a trustor accepts some level of untrustworthiness. Moreover, notable is that an experience

may be modelled as dogmatic or even absolute, expressing complete (unjustified) certainty. Again, such a model would need to approximate the motivation for trustworthiness in the first place.

Trust property 4: Trust is *transitive* (with restrictions)

There are suggestions against transitivity, i.e. of delegation of trust. With this we mean a perfectly normal (positive) trust delegation setting of *A* trusting *B* and say *B* trusts *C* who is to *A* previously unknown, see Figure 11 for illustration. In this case, the transitive relation is $((A \Rightarrow B) \wedge (B \Rightarrow C)) \Rightarrow (A \Rightarrow C)$. Arguments against such trust transitivity often point out that *A* did not trust *B* to delegate [62] [110], but trusted *B* to provide in the scope. If *B* does delegate despite this, it is called unintentional transitivity, i.e. *B* may have imposed restriction that *A* may not agree with or be aware of. For example, if *A* trusts *B* and *B* delegates to *C*, then *B* may have evaluated *C* based on qualifications that *A* does not agree with. This is prominent especially if the evaluation is subjective; how would *A* know that *B* evaluates *C* with the same sense of appreciation? Hence, trust transitivity is as if granting the trusted entity the power of deciding for the trustor.

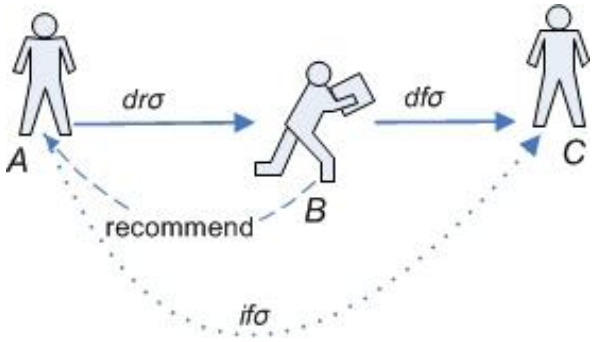

Figure 11: Trust transitivity

Dually to arguments against transitivity, arguments in favour of this are numerous. For example, the concept of reputation-based trust would boil down to second hand opinions without transitivity, i.e. as if instead of a reputation only asking friends for advice. To address this problem, the type of trust has been divided into *referral trust* and *functional trust* and a trust relation to *indirect* or *direct* trust. We illustrate this with the transitive relation above and in Figure 11 where *d* on an arc denotes direct, *i* indirect, *f* functional, *r* referral and σ trust. The question of whether or not *A* possesses indirect functional trust in *C* by direct referral trust in *B* is made subject to restrictions. These restrictions include that transitivity is valid only when the last leg of a relation is direct functional trust (i) [148] and all relations on the path share the scope of trust (ii) [148] [152]. Hence, for trust transitivity to hold, the trustor must explicitly rely on the trustee to delegate (i), i.e. for providing referral trust. In addition, if *A* possesses direct referral trust in *B* in recommending a *car mechanic*, then whomever *B*

recommends with direct referral or functional trust needs to share the scope (ii), i.e. be evaluated as *car mechanic*. Hence, trust transitivity requires matching scope. Moreover, transitivity with negative trust may have unwanted consequences, i.e. if *A* has negative trust in *B* and *B* negative trust in *C*, then nothing about *A*'s trust (or negative trust) in *C* may be derived. The notion of trust transitivity is elaborated on later in Section 4.3.2.

In addition to these properties on a general view of trust, additional properties have been suggested. These include that a trust relation is context-dependent, i.e. the scope as discussed with respect to the transitivity property.

Trust property 5: A trust relation is *context-dependent*

Whenever trust is experience-based, the level of trustworthiness evolves over time.

Trust property 6: Trust evolves over time

Hence, trust and the level of trustworthiness may change non-monotonically due to new experiences or lack of these.

In addition to these properties, each viable trust system ought to implement a representation of trust, a means to compute with it and means of setting the level of trust(worthiness). Hence, trust needs to be representable and measureable. The representation can be binary, discrete, based on continuous values or range; the computation can thereof be (i) logical, (ii) fuzzy, (iii) based on transitivity or (iv) probabilistic respectively. Existing implementations of these representations include (i) summation [1] [228]; (ii) REGRET [221]; (iii) PageRank [201]; and (iv) Bpdf [53] [142] [197], EigenTrust [155] respectively. EigenTrust is detailed in Section 4.3.1.1. The trust metric's scale can be of any kind but need to be partially ordered and is typically totally ordered, e.g. any real in [0, 1], {-1, 0, +1} with $-1 < 0 < 1$, {low, mediocre, high} where low $\leq$ medium $\leq$ high. With these scales, an interpretation of the outcome may be a threshold, rank as for greater the better, probability or mere cognition leaving it up to the human to decide [217].

4.1.2 Policy-Based Trust Systems

Policy-based trust has its roots in user authorisation, called *trusted computing* as defined by the Trusted Computing Group [114]. Essentially, this amounts to enforcing a given set of policies (rules) to determine a discrete level of trustworthiness. Sometimes this is called access control that is an example of a formal policy-based trust usually reduced to a Boolean decision [41]. For example, the combination of username – password provides certain rights to access a resource. Policy-based trust might also build up, called negotiation, when parties gradually reveal information in exchange for higher trust, e.g. TrustBuilder [260]. In fact, such negotiation is a strategy that gradually raises stakes in a manner that defecting is more costly than cooperating.

Artz and Gil [13] provide a survey on policy-based trust with more examples noting that having a sufficient policy, the recursive question of the policy for trusting the credentials arise; which is frequently solved by having a mutually trusted certification authority signing and verifying the credentials. An overview on such formal foundations for computational trust may be found elsewhere [163]. In the following, we provide the reader with an overview of the setting by outlining a few policy-based trust systems. We do this with the sole purpose of motivating our choice of experience-based trust for capturing trust in context.

4.1.2.1 Weeks' General Policy-Based Model

A general mathematical framework for modelling policy-based trust is presented by Weeks [249]. This model base on the least fixpoint in a (complete) lattice requiring a partially ordered set (policies) as well as monotonic functions on these. Consider the complete lattice (*Auth*, $\preccurlyeq$) and a set of entities *Principal*; where $\preccurlyeq$ denotes the binary order relation of the elements in *Auth*. If policy $a \preccurlyeq b$ when $a, b \in Auth$, then $a \preccurlyeq b$ means that policy *b* authorises at least as much as *a* in *Auth*. The lattice *Auth* specifies authorisations for a principal; here we consider $Alice \in Principal$. This authorisation is defined by a function *AuthMap* that maps *Principal* to *Auth*, i.e. $AuthMap = Principal \rightarrow Auth$ where *AuthMap* is a lattice under the pointwise extension as *Auth* is.

Consider the lattice to denote rights of *Alice*'s file access. Realistically, *Auth* = {N, R, W, RW} with the order relation N $\preccurlyeq$ R, N $\preccurlyeq$ W, R $\preccurlyeq$ RW, W $\preccurlyeq$ RW for N 'no right', R 'read', W 'write' and RW 'read and write'. Let $m_1 \in AuthMap$, then function m_1 describes the authorisations the principal(s) grants to *Alice*. For example, m_1 may be $m_1(Bob)$ = RW and $m_1(Claire)$ = R which means that *Bob* may grant *Alice* the right to RW and *Claire* may grant *Alice* to the rights to R. The license $l \in License$ is a monotone function $AuthMap \rightarrow_m Auth$, i.e. $(Principal \rightarrow Auth) \rightarrow_m Auth$ where $p \in Principal$ is authorised as specified by license $l(m)$. A licence $l(m)$ expressed in λ-calculus, e.g. $\lambda m.\sqcap\{W, m(Bob), m(Claire)\}$ means that the principal in question may "write if *Bob* and *Claire* may" as of the greatest lower bound ⊓ on the *Auth* lattice. That is, for a specific $m_1 \in AuthMap$, $m_1(Bob)$ = RW and $m_1(Claire)$ = R, then by reduction $\lambda m_1.\sqcap\{W, m_1(Bob), m_1(Claire)\}$ = N as *Claire* was not allowed to write.

They further define assertions $Assert = Principal \times License$ a pair $\langle p, l \rangle$ where $p \in Principal$ and $l \in$ License, read so that the issuer *p* authorises *l*. For example, $\langle Bob, \lambda m_3.RW \rangle$ asserts that *Bob* is assigned RW. The set of authorisations granted by *p* is $\{l(m) \mid \langle p, l \rangle \in A\}$ and its least upper bound $\sqcup\{l(m) \mid \langle p, l \rangle \in A\}$ describes a single most generous authorisation issued by *p*. The consistent authorisations are therefore the least fixpoint of the *AuthMap* lattice. Therefore, in the policy, the least fixpoint is whenever all principals agree and no changes in authorisations occur by iteration on the licenses. Examples of the fixpoint computations in the lattice can be found elsewhere [162] [249].

4.1.2.2 Other Notable Policy-Based Models

Other notable formal policy-based models include those of Fuchs et al. [100] [101], Carbone et al. [57] and Krukow [162]. Fuchs et al. [100] [101] strive to prove that some data actually origin from the source it is claimed, discarding its qualities but noting that this may be expressed in the framework as well. This is done by signature in a public key infrastructure guaranteeing security requirements. Carbone et al. [57] and Krukow [162] consider a trust structure $T = (D, \preccurlyeq, \sqsubseteq)$ where in addition to ordering trust values D by $\preccurlyeq$, also add another dimension, information ordering $\sqsubseteq$ stating that if $m \sqsubseteq m'$, then m' is based on more information (evidence). Let $D = \{unknown, low, mid, high\}$ then $low \preccurlyeq unknown \preccurlyeq high$ and $low \preccurlyeq mid \preccurlyeq high$ as well as $unknown \sqsubseteq low, mid, high$. Their goal is by defining T and a set of principals to find and establish a global trust state that represent each principal's trust in each other. This model may be used to define policies but seem to be restricted to access control [101].

The strength of these models are their drawback as well; as trust evolves and the autonomous environment changes, the common shortcoming shared by policy-based trust systems is that they employ a static form of interpretation on trust [55] [110] and do consider only exclusive and exhaustive matters. That is, in Weeks model [249] the simplest form of an update, a provider updating its policy (license) triggers a change in the related policies demanding a re-computation of at least a part of it [57], including the fixpoint. Moreover, the existence of a fixpoint is guaranteed only as long as the policy updating function is monotonic. Hence, it assumes non-revocation of rights and the universally agreed ordering of the lattice elements, e.g. an axiom stating that R is unrelated to W whereas in many cases, $R \preccurlyeq W$ and $W = RW$.

An alternative, but very interesting use of policy-based trust includes the reverse use of context and trust, called device comfort [179] [180]. Device comfort aims at providing the device a relative comfort by tasks that a user may want to perform. Whenever the comfort level is too low with respect to the task desired to perform, the device may refuse to perform a task or ask for additional authorisation. Obviously, contextual 'safe zones' may be used, e.g. home, office etc. Hence, device comfort seeks the device's comfort in performing a task in the context, e.g. if the device is not at work or at home, it may require further authentication for accessing e-mails.

Altogether, policies are what technology-driven mobile human-computer interaction has researched [159] [181] [199]. Traditionally its focus has been on the security aspects assuming non-functional requirements, such as availability, reliability, honesty [266]. Contrary, as the ubiquitous applications are increasingly performing tasks on behalf of its user [125], on means stated by the user [30] [79] and embedded in our everyday, a more user-centric approach is desired. Hence, policy-based trust as a framework for ubiquitous computing settings does not seem to fit very well and will not be considered further in this thesis.

4.1.3 Experience-Based Trust Systems

An experience-based trust system derives a level of trust based on past experiences on a trustee in a proposition. The concept of experience-based trust was coined by Barber [29] who defined three expectations of trust that experiences contribute to:

(i) an expectation of the fulfilment of the biological, physical and moral order persistence
(ii) an expectation of the technical competent role performance on the trustee and
(iii) an expectation on fiduciary obligations

For the expectations, (i) seeks for evidence of continuity and (ii) for evidence of competence whereas (iii) for evidence that the fiduciary will place the trustor's welfare above its own [28]. An example of a fiduciary obligation is the professional secrecy of a doctor on which a patient (trustor) places expectations. Falcone and Castelfranchi [93] further categorised the concept of trust into competence, disposition, dependence, fulfilment, willingness, persistence, self-confidence and motivation beliefs. All of these expectations are enforced by experiences.

Having a set of recorded experiences, whenever these are shared with other entities the system is a reputation-based trust system. Hence, a reputation-based system relies on first-hand experiences that typically are enforced by referrals' experiences, the reputation [220]. As each experience is a trustor's perception of a trustee, a reputation is subject to the perceiver's biases, making reputation-based systems very hard (if not impossible) to define formally [162]. Hence, recalling the discussion about transitivity and Figure 11, reputation-based trust is further divided to direct trust as for first-hand experiences and indirect trust for referrals' experiences, the reputation, also called *service provision* and *delegation* trust [110].

The difference between reputation and first-hand experiences is well shown by the following perfectly normal sentences [150]:

I trust you because of your good reputation.
I trust you despite your bad reputation.

The first sentence states trust based on the good reputation, i.e. in case of insufficient or inexistent first-hand experiences. The second sentence suggests that a trustor is in possession of some information that overweighs the bad reputation. Other factors that might influence trust are, among others, the contextual relation between the entities, called meta-knowledge [220], e.g. *mother_of*. Clearly this kind of relation is fundamental in the social trust.

Whenever the amount of first-hand experiences is insufficient, this gives rise to a level of uncertainty with respect to the level of trustworthiness. In case of uncertainty, for the trustor to ascertain a level of trustworthiness in a trustee in a proposition, reputation in form of referrals experiences may be inquired.

Composing such referral experiences, however, brings along several difficulties in the establishment of a level of trustworthiness. These include discounting of second-hand experiences and how to reach a consensus when several referrals are used. These are matters that the subsequent subsections will delve into.

4.1.3.1 Experience-Based Trust Levels

The level of trust in experience-based trust systems has been represented as both probabilistic and non-probabilistic. A probabilistic model outputs a percentage on the likeliness of a proposition whereas a non-probabilistic model typically a value lacking meaning, i.e. the greater the better. Examples of non-probabilistic trust models include EigenTrust [155], PeerTrust [262] and Abdul-Rahman's and Hailes' system [1]. Further, the level of trust in non-probabilistic systems may well be within [0, 1] and adhere to additivity, i.e., as if it was probabilistic. An example of this category is EigenTrust [155] that is presented in greater detail in Section 4.3. Below, we will elaborate on the probabilistic model.

Consider a probabilistic setting with a frame of discernment X of possible outcomes, called propositions, i.e. $X = \{x, \bar{x}\}$. Here $\bar{x}$ is the complement of x with $x, \bar{x} \in [0, 1]$ and $x + \bar{x} = 1$. Hence, this frame of discernment describes exclusive and exhaustive propositions in a binomial frame of discernment. Consider for brevity at the moment values of x (trustworthiness) and $\bar{x}$ (untrustworthiness) to be defined by the set of past experiences. Initially, an objectivist view with no experiences suggests a level of trustworthiness in a probabilistic model to indicate $x = 0.5$ and $\bar{x} = 0.5$, i.e. indicating equal probability. Similarly, with n-ary outcomes on a frame of discernment X, the initial equal distribution in a probabilistic model is $1/n$ where $n = |X|$ is motivated [210], i.e. all propositions of the frame of discernment are equally probable. Hence, such a view is unable to differentiate between uncertainty and certainty of no variance, i.e. no evidence and full evidence of equal distribution. To exemplify this, consider a sealed box containing *red*, *green* and *blue* balls; initially the probabilistic model is indifferent from that of having 12 experiences with 4 of each colour when discarding experience dissolving by time, typically called aging.

This raises the compelling need to express uncertainty as opposed to certainty, i.e. a subjectivist view on probabilities. Here, uncertainty must not be confused with 'untrustworthy' [57] [177] as untrustworthiness refers to evidence of 'not trustworthy' and uncertainty refers to the lack of evidence, i.e. 'do not know'. Hence, trustworthy is opposed to untrustworthy and certainty is opposed to uncertainty; the level of evidence is related to the experiences. The importance of the concept of uncertainty is further emphasised in scenarios with incomplete information. Such scenarios include, but are not limited to, scenarios where the decay of experiences as a function on time or inherent inaccuracy on the acquired information is applied. Consequently, we conclude that this kind of

trustworthiness is what the inherently imperfect context derived in an autonomous architecture yearns for. As of this, for the sealed box with coloured balls, the initial trustworthiness and untrustworthiness in the ball being *red* is necessarily 0, as is the case for all other colours as well. This is because there is no experience giving rise for any certainty in any proposition; hence, the uncertainty is 1. To represent this, Dempster-Shafer theory seem to qualify well.

4.1.3.2 Dempster-Shafer Theory

Dempster-Shafer theory represented by a Belief function is a generalisation of the Bayesian theory of subjective probability as Belief functions allow uncertainty on the power set of propositions [144]. Its domain is a set of outcomes *X* where the mass (certainty) *m* denotes the evidence of each and *m*: $2^X \rightarrow [0, 1]$. The probabilistic view on the evidence assigns a mass *m* to each element in 2^X and is called basic belief assignment where $m(\emptyset) = 0$ and $\sum_{x \in 2^x} m(x) = 1$ assuming that there is an outcome every time, i.e. in case of the sealed box assuming a ball and not a cube is picked each time. Hence, the possible outcomes conform to additivity. This additivity is modelled on a mass space, e.g. *Ball* = {*red*, *green*, *blue*} then the mass '*red* or *green*' denote the certainty of a ball not being *blue*, but not certain whether it is *red* or *green*; realistically the case when a red – green colour blind person is performing the evaluation.

In addition to the mass *m*, disjoint sets of probabilities *bel* are defined $bel(A) = \sum_{B \subseteq A} m(B)$, i.e. the sum of the masses of its subsets. Hence, the belief denotes the 'certainty' or 'evidence in' the proposition, e.g. $bel(\{red, green\}) = m(\{red\}) + m(\{green\}) + m(\{red, green\})$. A feature is that the mass of the total set $m(Ball) \neq 0$ is reasonable in case of a blind person evaluating but $bel(Ball)$ is 1 due to additivity. Plausibility *pl* denotes the 'max probability' or that 'there is evidence against this proposition to a level' where $pl \geq bel$ and $pl(A) = \sum_{A \cap B \neq \emptyset} m(B)$, the sum of non empty intersecting masses; or more conveniently, $pl(A) = 1 - bel(\bar{A})$ where $\bar{A}$ denotes the complement of *A*, e.g. $pl(\{red, green\}) = m(\{red\}) + m(\{green\}) + m(\{red, green\}) + m(\{red, blue\}) + m(\{blue, green\}) + m(\{red, green, blue\})$ or, equivalently, $1 - bel(\{blue\})$. With mass, belief and plausibility, intervals may be expressed within this framework, where plausibility and belief denote the upper and lower limits respectively. Uncertainty is the interval between *pl* and *bel*, the probability lacking evidence in favour for or against the proposition.

Further notable is that Dempster's rule of combining independent evidence has been criticised as providing counterintuitive results when combining conflicting evidence [273]. This gives rise to a number of combination operations addressing this shortcoming [231] and raises discussions on its domains of applicability [204]. Pearl [204], however, notes that belief theory is a theory on the probability of provability as opposed to probabilities of truth, i.e.

that belief theory may be used to derive on the certainty. That is, "*Bel*(*A*) stands for the probability that the constraints imposed by the available evidence, together with the constraints that normally govern the domain, will be sufficient to compel the truth of *A* and exclude its negation" [204], i.e. that *Bel*(*A*) provides a certainty level of the truth of *A*. However, as examining Dempster's rule of combination and arguing for and against it is out of the scope of this thesis, we will not discuss this matter any further.

4.1.3.3 A General Model for Representing Trust

To express the levels of trust, we use a general representational model for experience-based trust. The model is inspired by Krukow's general model [162]. In this model, an experience *Exp* the trustor $P \in \{<Entities>\}$ has recorded is modelled as a 4-tuple: $Exp^P = (\delta, \epsilon, \zeta, \eta)$ where $\delta \in \{<Entities>\}$ is the *trustee* and is a long term identification, ϵ is the *datum* of interest where $\epsilon \leq \epsilon_0$ and ϵ_0 denote a specific perspective taken where the subscript defines the interval with $_0$ denoting 'now', $\zeta \subseteq \{<Propositions>\}$ and $\eta \in \{<Score>\}$. The datum ϵ is typically time, but other continuous data may also be considered, e.g. sociality. The *Score* is the trust metric's scale and is considered hereafter totally ordered, recall Section 4.1.1. The implemented type of the metric is trivial for modelling and becomes relevant only when calculating.

With this model of representation, an entity's history of experiences is a set of experiences, $Exp^P = \{(\delta, \epsilon, \zeta, \eta)\}$. As of this, $Exp^P(\epsilon_0) = \{(\delta, \epsilon, \zeta, \eta)\}$ and $Exp^P(\epsilon_i) = \{(\delta, \epsilon \leq \epsilon_i, \zeta, \eta)\}$ as inquiring for history prior to ϵ_i. Hence, writing $Exp^P(R, \epsilon_0)$ calls for a set of direct experiences trustor *P* has had with trustee *R* where $Exp^P(R, \epsilon_0) \subseteq Exp^P(\epsilon_0)$ and $Exp^P(R, \epsilon_0) = \{(R, \epsilon, \zeta, \eta)\}$. Writing $Exp^P(R, \epsilon_0, \zeta) = \{(R, \epsilon, \zeta, \eta)\}$ provides the set of experiences regarding a trustee R in a proposition ζ at datum ϵ up until ϵ_0 whereas for a specific datum, $Exp^P(R, \epsilon, \zeta) = \eta$ provides a score. Dually, we may write $Exp^P(P, \epsilon_0, \zeta) = \{(P, \epsilon, \zeta, \eta)\}$ for the experience *P* has had in itself in the proposition ζ.

In text, we acknowledge the out of the ordinary use of capital letters as a single entity of a set. Lower case letters are provided a special meaning of their own when representing trustworthiness. For the *Propositions*, $\zeta \subseteq \{<Propositions>\}$ as a composition of outcomes may be of interest; a matter that is of ontological nature and thereof not considered further.

4.1.3.4 Reputation on the General Model of Trust

The general model representation of experiences provides the basis for storing them and hence, deriving referrals' opinions as reputations. In this case, reputation is, for example, when trustor *P* ascertains $Exp^P(R, \epsilon_0)$ by a referral's experiences in *R*, e.g. by *Q*'s experiences in *R*: $Exp^Q(R, \epsilon_0)$ where $P \neq Q$. We

omit modelling the "request", i.e. the message sent from *P* to *Q* for $Exp^Q(R, \epsilon_0)$. Obviously, *P* may possess experiences $Exp^Q(R, \epsilon_m)$ whose applicability is defined by ϵ_m age. This forms the need of transitivity in accordance to trust property 4, i.e. $Exp^Q(R, \epsilon_0, \zeta')$ should be discounted by *P*'s level of referral trust in *Q* in ζ for proposition ζ'. Moreover, we assume in accordance to principles for ubiquitous systems that the experiences are stored in a distributed manner. In a system storing the experiences centrally, the "global trust" of *R* would rudimentary be calculated from $\{\bigcup_{i=1}^{n} Exp^{P_i}(R, \epsilon_0)\}$ when $P_i \in Entity \setminus R$ and $i = 1, \ldots n$ and $n = |Entity \setminus R|$. In this case, any composition of experiences according to demands may be used to calculate a trust level, much as it is done in Wikipedia reputation systems [160].

In case of referral experiences as above, when referring to entity *Q* in a distributed environment, *Q* is faced with the decision of trusting *P* with possibly sensitive detailed information. This gives rise for *Q* to consider hiding information for the sake of preserving privacy and intimacy. This is done by the abstracting experience operator *Abs*; we write $Abs\left(Exp^{\delta}(\epsilon_0)\right)$, more precisely in this case $Abs(Exp^Q(R, \epsilon_0, \zeta'))$. As means to accomplish such abstraction, calculation on the score of the distinct experiences is demanded. This requires defining this score that we postpone to Section 4.3.2 and ask the reader for the moment to consider abstraction merely as a composition of the scores omitting the timestamp, i.e. $Abs\left(Exp^{\delta}(\delta', \epsilon_0, \zeta)\right) = \left(\delta', \epsilon_0, \zeta, \sum_{Exp^{\delta}(\delta', \zeta)} \eta\right)$. We define the function *fourth* as the fourth-component projection, i.e., projecting on the score of the tuple: $fourth(\delta, \epsilon, \zeta, \eta) = \eta$.

Whenever a trustee is referred to by a trustor for experiences in a scope in a third entity, this gives rise for trustor to serially compose each acquired $Exp^{\delta}(\delta', \epsilon_0, \zeta')$ or the $Abs\left(Exp^{\delta}(\delta', \epsilon_0, \zeta')\right)$ with the trustor's trust in the trustee, i.e. if *Q* is referred to by *P* for its experiences in *R*, then $Exp^Q(R, \epsilon_0, \zeta')$ or the $Abs\left(Exp^Q(R, \epsilon_0, \zeta')\right)$ is to be discounted by $Abs\left(Exp^P(Q, \epsilon_0, \zeta)\right)$. Hence, this calls for parallel and serial composition, i.e. $Abs\left(Exp^P(R, \epsilon_0, \zeta')\right) \blacksquare \left(Abs\left(Exp^P(Q, \epsilon_0, \zeta)\right) \square Abs\left(Exp^Q(R, \epsilon_0, \zeta')\right)\right)$ where $\square$ is a placeholder for serial composition and $\blacksquare$ a placeholder for parallel composition. Such combination forms the first link of a network of trust where *Q* is called a referral of trustor *P* in deriving the level of experience-based trustworthiness in trustee *R*.

4.1.4 Networks of Trust and Derivation Graphs

A network of trust is formed when two or more entities collaborate by sharing experiences regarding a trustee in a proposition, e.g. *P* collaborating with *Q* to ascertain its level of trust on *R* in ζ. As *P* may not trust *Q* in ζ, but only as a

referral as in recommending another entity for ζ, the distinction between direct referral trust and direct functional trust of P trusting R in ζ is necessary.

The structure of such a trust network is a directed graph $G = (V, E)$. Graph G may be cyclic, e.g. *Alice*'s trust in *Bob* must not prohibit *Bob* from expressing trust in *Alice*. However, for each instance of trust derivation, the path(s) needs to be acyclic. Such a graph is a directed acyclic graph (DAG) where a derivation path is denoted ρ. These paths are the chains of trust that are calculated with.

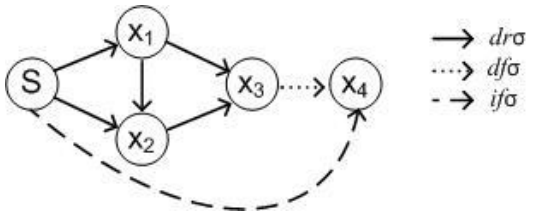

Figure 12: DAG not being a DSPG

Considering the conditional dependency structure of Figure 12 with three valid paths denoted 'ρ' from trustor S to trustee x_4. Let ';' denote serial composition, ⋄ parallel and *f* functional, *r* referral, *d* direct, *i* indirect and σ trust in accordance to [148] [151]. Then the paths are:

$\rho_1 = [S, x_4, if\sigma] = [S, x_1, dr\sigma] ; [x_1, x_3, dr\sigma] ; [x_3, x_4, df\sigma]$
$\rho_2 = [S, x_4, if\sigma] = [S, x_2, dr\sigma] ; [x_2, x_3, dr\sigma] ; [x_3, x_4, df\sigma]$
$\rho_3 = [S, x_4, if\sigma] = [S, x_1, dr\sigma] ; [x_1, x_2, dr\sigma] ; [x_2, x_3, dr\sigma] ; [x_3, x_4, df\sigma]$

Common to all is that the last arc in the path is direct functional, noted as a condition for transitivity in Section 4.1.1, and all the other arcs are referral trust, all sharing the same proposition [152]. The different compositions of the indirect functional trust from S to x_4 are $2^{\{\rho1,\, \rho2,\, \rho3\}} \setminus \emptyset = \rho_1 \mid \rho_2 \mid \rho_3 \mid \rho_1 \diamond \rho_2 \mid \rho_1 \diamond \rho_3 \mid \rho_2 \diamond \rho_3 \mid \rho_1 \diamond \rho_2 \diamond \rho_3$. Yet, the parallel composition of $\rho_1 \diamond \rho_2$ can yield two configurations i.e. $([S, x_1, dr\sigma] ; [x_1, x_3, dr\sigma] ; [x_3, x_4, dr\sigma]) \diamond ([S, x_2, dr\sigma] ; [x_2, x_3, dr\sigma] ; [x_3, x_4, df\sigma])$ and $(([S, x_1, dr\sigma] ; [x_1, x_3, dr\sigma]) \diamond ([S, x_2, dr\sigma] ; [x_2, x_3, dr\sigma])) ; [x_3, x_4, df\sigma]$. These configurations provide different output whenever ';' and '⋄' are not considered binary 'AND' and 'OR'; a problem identified by Jøsang in 1999 [145] noting that either some evidence is discarded or the independence is violated.

To resolve this, a restriction that each arc must only appear once in each set of derivation paths is introduced, called canonical expression [148]. Such a restriction defines the latter configuration of $\rho_1 \diamond \rho_2$ parallel configuration correct [151], i.e. where $[x_3, x_4, df\sigma]$ appears only once. Moreover, this restriction makes the derivation graph a Directed Series Parallel Graph (DSPG). A DSPG may be constructed by applying the following series and parallel rules on $G = (V, A)$ with $S, x_4, u \in V$ [98]:

Series: replace the arc (S, x_4) with (S, u) and (u, x_4) where u is a new vertex.
Parallel: replace the arc (S, x_4) with two arcs $(S, x_4)_1$ and $(S, x_4)_2$

Obviously, the parallel arcs $(S, x_4)_1$ and $(S, x_4)_2$ are disjoint only when applying series composition on either or both. In trust derivation, these rules establish canonical paths ρ between trustor and trustee, *s* and x_4. In addition, each DSPG is a special case of a DAG, i.e. the DAG in Figure 12 is not a DSPG but removing (x_1, x_2) or (S, x_2) invalidating ρ_3 or ρ_2 respectively makes it a DSPG. Composing parallel paths each providing a set of experiences therefore increase the level of certainty. Contrary, sequential composition that makes the paths 'longer' suggests less certainty.

4.2 Problem analysis

Trust and trustworthiness in the context of this thesis are issues that stem from the uncertainty on the data provider's capability in supplying correct data, i.e. the acquired data's probability of provability. Its metric is motivated to include uncertainty as opposed to certainty of (un)trustworthiness. Moreover, as of the decentralised setting of inconsistently behaving autonomous entities, trustworthiness builds up from initial uncertainty by local and referral experiences and changes continuously in a non-monotonic manner. This motivates the trustor and the user application to continuously monitor and measure trustworthiness, leading us to stating a Challenge 5.

Challenge 5 A means to calculate with trustworthiness for monitoring data reliability.

Addressing Challenge 5 requires the ability to calculate with possibly conflicting experiences. Moreover, the unpredictable behaviour suggests a decay of experiences according to the recorded context datum. This decay facilitates prompt reaction to a change in the behaviour of a trustee or referral.

4.3 State of the art

In experience-based trust an experience is optimally a realisation of a subjective perception by cognition [64] of the trustor and contributes to the level of certainty. In case of insufficient certainty, an entity may ascertain its level of trustworthiness by referral's experiences in the trustee, the reputation-based model. Computational models for reputation-based trustworthiness can be divided by their representation into probabilistic and non-probabilistic models. For probabilistic approaches, the outcome is a percentage whereas for non-probabilistic, typically 'the greater the better'.

In this section we clarify the differences between these. Moreover, we outline Subjective Logic that is a framework able to represent and calculate with uncertainties. Whenever the algorithms are not self-explanatory, examples are provided.

4.3.1 Non-Probabilistic Trust Computation Models

The non-probabilistic computational models for calculating of trustworthiness provides trustworthiness as a value without uncertainty [1] [14] [115] [155] [201] [228] [262]. Methods implementing a non-probabilistic model include methods aiming to add evidence [1] [201] [228], average on scores already in the closed interval [0, 1] [14] and normalisation of the score [115] [155] [262]. In these, the semantics of the non-probabilistic model's output is typically 'the greater the better'. This drawback may be illustrated by considering a score $\eta \in [0, 1]$ of binomial experiences. Moreover, let $P, Q, R \in Entities$ and P trusting Q be denoted $T_{PQ} \in [0, 1]$. Then, if the outcome of the applied method on experiences $Exp^{\delta}(\delta', \epsilon_i, \zeta)$ in a non-probabilistic system for entities yields $T_{PQ} = T_{PR}$ it is possible to say that from entity P's view, entities Q and R are equally trustworthy. However, the semantics provide no means to tell how trustworthy, or how certain the indicated posterior expectation value is or on how extensive evidence this score is based on and what the distribution is, as is the case of EigenTrust [155]. Dually, if $T_{PQ} = 0.4$ and $T_{PR} = 0.6$ the semantics merely supports a conclusion that $T_{PQ} < T_{PR}$. That is, the detailed information is lost during abstraction and aggregation of $Exp^{P}(Q, \epsilon_i, \zeta)$ to T_{PQ} and one can only tell that the greater the better. To the best of our knowledge, this drawback is similar for all such approaches that are discussed in Section 4.1.2. Thus we will for brevity only outline the seminal algorithm of EigenTrust [155] in greater detail to motivate our selection of a probabilistic model. Probabilistic models are examined in Section 4.3.2.

4.3.1.1 EigenTrust Explained

In EigenTrust [155], each experience is rated either unsatisfactory or satisfactory, making the score binary $\eta \in \{0, 1\}$. Consider entities i and j in line with Kamvar et al. [155] and the sum of the satisfactory experiences as $sat_{ij} = fourth\left(Abs\left(Exp^{\delta}(\delta', \epsilon_0, \zeta)\right)\right)$. Moreover, with the binary score, consider the complement of the abstracted score $Abs^{-}\left(Exp^{\delta}(\delta', \epsilon_0, \zeta)\right) = \left(\delta', \epsilon_0, \zeta, \sum_{Exp^{\delta}(\delta', \zeta)} 1 - \eta\right)$ that defines unsatisfactory experience $unsat_{ij} = fourth\left(Abs^{-}\left(Exp^{\delta}(\delta', \epsilon_0, \zeta)\right)\right)$. The abstracted score s_{ij} of entity i regarding entity j is:

$$s_{ij} = sat_{ij} - unsat_{ij}$$

The score s_{ij} loses critical details by composing the history of propositions to one irreversible metric. This is correctly noted by [155] as that s_{ij} of an entity with

poor experience is the same as for no experience, e.g. s_{ij} is the same for $sat_{ij} = 1$ and $unsat_{ij} = 0$ and $sat_{ij} = 10$ and $unsat_{ij} = 9$.

The abstracted score s_{ij} is normalised for entity j with respect to all other entities s_{ik} where $k \in Entities \setminus i$, i.e. entities that i may have had direct experience with. The normalisation is intended to countermeasure arbitrary high influence of one entity's experiences. However, for newcomers lacking any experiences, a set of pre-trusted entities P are provided with initial trust of $p_j = 1 / |P|$ when $p_j \in P$ and $p_j = 0$ otherwise. With this, the normalised local trust value c_{ij} is:

$$c_{ij} = \begin{cases} \frac{\max(s_{ij}, 0)}{\sum_k \max(s_{ik}, 0)} & if \sum_k \max(s_{ik}, 0) \neq 0 \\ p_j & otherwise \end{cases}$$

For example, let $Entities = \{x, y, z\}$ and from entity y's point of view $sat_{yx} = 8$, $unsat_{yx} = 2$, $sat_{yz} = 3$ and $unsat_{yz} = 0$, then $c_{yx} = ⅔$ and $c_{yz} = ⅓$ where the sum is 1.

Aggregating the local trust value c_{ij} with known referrals' trust values defines an entity's extended view of the environment.

$$t_{ij} = \sum_k c_{ik} c_{kj}$$

To motivate this fundamental view, consider the three entities, x y and z in this order with the following normalised experiences in each other:

$$c_{xj} = (0, 1, 0), c_{yj} = \left(\frac{2}{3}, 0, \frac{1}{3}\right), c_{zj} = \left(\frac{1}{8}, \frac{7}{8}, 0\right)$$

That by a i-by-j global matrix $C = [c_{ij}]$ is:

$$C = \begin{bmatrix} 0 & 1 & 0 \\ \frac{2}{3} & 0 & \frac{1}{3} \\ \frac{1}{8} & \frac{7}{8} & 0 \end{bmatrix}$$

Then deriving t_{ij} for each is as if asking friends (referrals); calculating t_{ij} for this example gives after one iteration:

$$t_{xj} = \left(\frac{2}{3}, 0, \frac{1}{3}\right),$$
$$t_{yj} = \left(\frac{1}{24}, \frac{23}{24}, 0\right),$$
$$t_{zj} = \left(\frac{14}{24}, \frac{3}{24}, \frac{7}{24}\right)$$

The result is the same as of C and vector $\vec{t}_i = C^{\mathrm{T}} \vec{c}_{ij}$ denoting each entity's opinion, i.e. C transposed times $\vec{c}_{ij}$, e.g.

$$\vec{t}_y = C^T * c_{yj} = \begin{bmatrix} 0 & \frac{2}{3} & \frac{1}{8} \\ 1 & 0 & \frac{7}{8} \\ 0 & \frac{1}{3} & 0 \end{bmatrix} * \left(\frac{2}{3}, 0, \frac{1}{3}\right) = \left(\frac{1}{24}, \frac{23}{24}, 0\right)$$

The matrix C denotes on each row the trustworthiness an entity perceives in other entities; when transposed, this denote on the row the other entities' trust in one entity, e.g. row 1 in the example C^{T} denotes the trustworthiness others have in x. Obviously, the sum of the entries in the vectors $\vec{t}_i$ adds up to 1 meaning that additivity of each entity's opinion is preserved.

Having the normalised satisfactory and unsatisfactory experiences represented as C^{T} denoting trust after asking friends, asking friends of friends propagates on the network of trust providing more referral evidence. This is performed by multiplying C^{T} by itself $(C^{\mathrm{T}})^{\mathrm{n}}$. Deriving matrix $(C^{\mathrm{T}})^3$ is shown below, i.e. the result after asking friends of friends.

$$(C^T)^2 = \begin{bmatrix} 2/3 & 1/24 & 14/24 \\ 0 & 23/24 & 3/24 \\ 1/3 & 0 & 7/24 \end{bmatrix}, (C^T)^3 = \begin{bmatrix} 368/576 & 39/576 & 325/576 \\ 24/576 & 529/576 & 90/576 \\ 184/576 & 8/576 & 161/576 \end{bmatrix}$$

With sufficiently large n, $\vec{t}^n = (C^T)^n \vec{c}$ basically converges to a global trust value $\vec{t}^n$ that is the eigenvector; in this example $\vec{t}^n \approx (0.163, 0.348, 0.489)$ when n = 145. We note that convergence is with some tolerance whose accuracy increases with n, hence, ≈. This also proves the calculations on C irreducible and aperiodic, i.e. there is no void data and no cycles in the values.

In the distributed version of EigenTrust, i.e. when the experiences are stored locally on each entity, the trust vector $\vec{c}_i$ in addition to its global trust value t_{i} is calculated by:

$$t_i^{(k+1)} = (1-a)\left(c_{1i}t_1^k + \ldots + c_{mi}t_m^k\right) + ap_i$$

Here a is the frequency of selecting a non-trusted entity, $c_{1\mathrm{i}}$ is entity 1 local normalised trustworthiness in i and t_1^k is the first entity's global trust value. They correctly note this not to be computationally very expensive as many $c_{ji}t_j^k = 0$. They continue to present how the managers of the experiences may be distributed securely using distributed hash tables (DHT), with the assumptions of robust and well-designed DHTs. These assumptions include that an experience manager does not tamper and successfully passes the values to a "live" entity when leaving the system, i.e. no redundancy and synchronisation is considered. Moreover, they provide personalising by biasing local experience $\vec{c}$ over the global $\vec{t}$ that is achieved by a constant d in the interval [0, 1].

$$\vec{t}_{personal} = (1-d)\vec{c} + d\vec{t}$$

Critics regarding EigenTrust include the treatment of newcomers, unsatisfactory reputation rated as 0, no model for aging, intermediaries do not get recognised, the information s_{ij} is critically abstracted, and the score is relative to the selection of the pre-trusted entities. While all other points of criticism may be considered features of the algorithm, the selection of pre-trusted entities is essential [155] and forms a critical single point of failure [140] as the pre-trusted entities determine the set of peers by which the entity will start interacting with.

This has been verified by simulations [255]. Hence, if one of these pre-trusted entities subvert to malevolence, this jeopardises the whole system. Moreover, as matrix multiplication is computationally costly, any update or decay (such as by time) generates a new matrix by each logical hop. Critics regarding the fundamental assumption that each element of the matrix has an opinion at all imply EigenTrust to regard bi-directional trust where *Alice* trusting *Bob* implies *Bob* to express his trust in *Alice*. In addition, as mentioned, EigenTrust suffers from the lack of semantics with the mere interpretation of 'the greater the better', noted by the authors [155]. These are also matters that, to the best of our knowledge, all non-probabilistic trust systems suffer from.

Related to EigenTrust is the model presented by Guha et al. [115] with the difference of considering trust and distrust matrices in separation to predicting the unknown value. They motivate their approach, with respect to among others EigenTrust, by that expressing distrust is equally important, where the score of 0 may be confused between '*don't know*' and '*don't trust*'. They also propose some new types of trust propagation, namely direct propagation, co-citation, transpose trust and trust coupling. Of these, direct propagation and co-citation may be relevant for contexts. In direct propagation, if *A* trusts *B* then whatever *B* trusts, *A* is considered to trust as well; in terms of matrices this is expressed as $A' = A \times A$. In co-citation, if *A* trusts *C* and *D* and *B* trusts only *C*, then by co-citation *B*'s trust in *D* may be derived; in terms of matrices this is expressed as $A' = A \times A^{T} \times A$, e.g. who trusts the same entities as *B* will imply *B* to trust those as well. The critics for EigenTrust are valid for Guha et al. [115] framework as well though they state that their initial matrices are given. Hence, they abstract among others, the critics regarding newcomers and pre-trusted entities, but do not solve them.

4.3.2 Computational Models for Probabilistic Trust

A probabilistic trust model represents trust as a probability with an output $0 \leq x \leq 1$. Examples of computational models for probabilistic models include maximum likelihood by Despotovic and Aberer [2], TrustNet [272] based on Dempster Shafer theory and Bayesian models based on statistical updating of Beta Probability Density Functions (Bpdf) as of spanning [0, 1] interval [53] [142] [149] [197] [244]. Of the probabilistic systems, the ones based on Dempster Shafer theory capturing uncertainty seem the most versatile. However, these systems often lack a representation of trustworthiness as they omit considering how or what is an expression of (dis)trust, i.e. what are the input values composed of. Bpdf models are used for representation where evidence (experiences) is denoted as a tuple (α, β). In the tuple, α denotes experience of satisfactory behaviour and β denotes experience of unsatisfactory behaviour; hence, very similar to *sat* and *unsat* of EigenTrust. Moreover, Bpdf

represent uncertainty by distribution with complete uncertainty (no experience) as even distribution.

Including a degree of uncertainty, however, demands deciding on the level of certainty needed for triggering an actuator. It also motivates acquiring referrals' experiences to increase the level of certainty [53]. However, these referral experiences need to be discounted by the trustor's trustworthiness in the referral. Hence, a sufficient computational model needs to manage sequential and parallel transitivity as well as combining disjoint sources in a mathematically sound manner including the properties listed in in Section 4.1.1 from easily expressible experiences. These criterions are met by Subjective Logic, hence motivating this for more detailed presentation.

4.3.2.1 Subjective Logic Framework

Subjective Logic is a probabilistic logic that addresses uncertainty, provides a means of transitivity and derives a level of subjective belief in an entity in a proposition [141] [142] [144] [150]. Moreover, it provides a computational model for calculating with trustworthiness. Subjective Logic is related to Dempster-Shafer theory and consists of logical operators; it is also related to Bpdf as there is a unique transformation rule (shown shortly) and it may be used to analyse Bayesian networks. Hence, Subjective Logic is both belief-based and Bayesian as a Bayesian update (the posterior adding evidence) is straight forward [150], presented in Section 4.3.2.5. Hence, Subjective logic provides a viable model for calculations on the QoC parameter of trustworthiness. Moreover, we cite Jøsang that "Subjective logic must not be confused with fuzzy logic. The latter operates on crisp and certain measures about linguistically vague and fuzzy propositions, whereas the subjective logic operates on uncertain measures about crisp propositions" [141]. Here crisp is used as a substitute for lack of uncertainty. This means that as fuzzy logic operates on fuzzified crisp values and fuzzy propositions, the subjective logic operates on values with uncertainty on a certain proposition.

The trustworthiness "type" in a Subjective Logic is an opinion, denoted ω. Opinion ω_x^A denotes the opinion held by an entity A in proposition x. When sequential reasoning is utilised, $\omega_x^{A;B}$ denotes A having an opinion on B in proposition x. An opinion ω is always expressed on a binomial proposition, e.g. binomial: $ball \in \{colour_1, \neg colour_1\}$; recall 4.1.3.1 and 4.1.3.2. Moreover, the Subjective Logic is a generalisation of binary logic; meaning that whenever an opinion is Binary, the Subjective Logic operators behave alike their corresponding logical expressions [144]. Obviously, Subjective Logic also scale to multinomial opinions, i.e. n-ary $ball \in \{colour_1, colour_2, \ldots, colour_n\}$. The following subsections define an opinion and means to calculate with the recorded experiences to acquire the momentarily subjective level of trustworthiness as perceived by an entity.

4.3.2.2 Parameters of an Opinion

The representation of trustworthiness as subjective opinions on frames of discernments as presented in this section follows that of Jøsang [144]. There, the representation of a multinomial subjective opinion is by a belief vector $\vec{b}$, an uncertainty scalar *u* and a base rate vector $\vec{a}$ in a *k*-nomial barycentric coordinate system. Assume a frame of discernment $Q = \{q_i | \ i = 1, \ldots, k\}$ where $k = |Q|$ and $\bigcap q_i = \emptyset$, i.e. a frame of discernment on a finite number of outcomes that are exclusive and exhaustive, e.g. picking a ball of a certain colour $q_i \in Q$ from a box. The belief mass vector for an outcome q_i is $\vec{b}(q_i)$ where $\sum_{q \in Q} \vec{b}(q) \leq 1$ and $\vec{b}(\emptyset) = 0$. That is, $\vec{b}(q_i)$ denotes the belief in outcome q_i whose sum is subadditive as of uncertainty. Uncertainty scalar *u* is defined $u = 1 - \sum_{q \in Q} \vec{b}(q)$, i.e. $u \in [0, 1]$. To acquire an *expectation value* with $u = 0$, a base rate vector of non-informative a priori probability is introduced. This base rate vector on each outcome is defined $\sum_{q \in Q} \vec{a}(q) = 1$ where $\vec{a}(\emptyset) = 0$. The expectation value vector is defined $\vec{E}_Q(q_i) = \vec{b}(q_i) + \vec{a}(q_i) * u$.

Having these vectors in a *k*-nomial barycentric coordinate system on a frame of discernment *Q*, the composite function over *Q* is $\omega_Q^P = (\vec{b}, u, \vec{a})$. It denotes *P*'s opinion on *Q* where $\vec{b}$ and $\vec{a}$ have *k* parameters each and *u* is a scalar. Hence, the multinomial opinion will have $2k + 1$ parameters. With the opinions, the subscript indicates the frame of discernment and the superscript the owner of this opinion. We may omit expressing the owner when trivial.

4.3.2.3 Representing the Trustworthiness as Opinions

In a representation of trust, all the properties of trust mentioned in Section 4.1.1 needs to be addressed. One of these is that of the incompleteness of trusts, i.e. complete certainty cannot exist indicating that $u > 0$. The base rate vector $\vec{a}$ is therefore always influential in finding the expectation value. Moreover, n-ary *Q* over exclusive and exhaustive outcomes is easily coarsened to a binary view by defining $\vec{b}(\bar{q}_i) = \sum_{q \in Q/q_i} b(q)$ and $\vec{a}(\bar{q}_i) = \sum_{q \in Q/q_i} a(q)$ reducing the cardinality of the set of outcomes $|Q| = 2$, i.e. to a binomial proposition $Q = \{\bar{q}_i, q_i\}$. However, viewing this proposition in Dempster-Shafer theory, the belief mass *bel* is $m(q_i)$, uncertainty mass $m(\{q_i, \bar{q}_i\})$ from which *pl* may be defined as *bel* + *u* or $m(q_i) + m(\{q_i, \bar{q}_i\})$. Hence, disbelief is mass $m(\bar{q}_i)$, i.e. evidence against q_i [153].

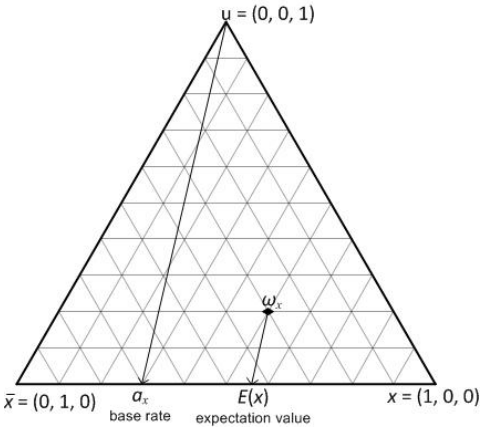

Figure 13: Binomial opinion triangle on a binary frame of discernment

A binomial opinion may be illustrated by a 2+1 vertex shape such as a triangle depicted in

Figure 13 [145] formed by vertices u, x and $\bar{x}$, trinomial by a tetrahedron formed by u and exclusive and exhaustive vertices $x \in Q$ where $|Q| = 3$. Similarly, an n-nomial opinion on an n-nary frame of discernment may be depicted by an n+1 vertex shape, i.e. by a Dirichlet Probability Density Function (Dpdf) [147]. Any area coordinate (point) in an n-nomial barycentric coordinate space adheres to additivity and is given by an n+1 tuple. Moreover, as noted, any n-ary frame of discernment of exclusive and exhaustive outcomes may be coarsened to a binary view.

As an opinion is binomial, it is written $\omega_x^A = (b_x^A, d_x^A, u_x^A, a_x^A)$. This opinion exempt of the base-rate a_x is a point in a triangle and its area coordinate is formed by vector $\vec{b}(q)$ denoting b_x^A, scalar u denoting u_x^A, a_x^A as for base-rate and d_x^A derived from vector $\vec{b}(\bar{q})$. Dually, a tetrahedron is formed by 4 vertices and an area coordinate is defined as a 4-tuple.

An expectation value on the binomial view is denoted $E(x)$ and defines the apriori assumed distribution of the uncertainty u [145], calculated:

$E(x) = b_x + u * a$

Thus, an expectation value is a point on the basis of the triangle. The expectation value proves its importance when ordering opinions in a total order based on belief. Otherwise, deciding whether $\omega_x \leq \omega_y$ or $\omega_x \geq \omega_y$ for arbitrary propositions x and y for example with opinions ω_x = (0.3, 0.3, 0.4, a) and ω_y = (0.4, 0.4, 0.2, a) is impossible as ω_y depicts more trust, but more distrust as well.

4.3.2.4 Subjective Logic on the General Model

Modelling experiences of the general model as opinions of subjective logic requires a means of composing the disjoint $Exp^{\delta}(\epsilon_0)$ to one abstract experience. Such an abstract experience qualifies for referral experiences. It should however take into account possible experience-specific characteristics, for example, aging of experiences. Hence, it provides some privacy and intimacy by hiding details, as noted in Section 4.1.3.4.

The abstracted experience $Abs\left(Exp^{\delta}(\delta', \epsilon_0, \zeta)\right)$ is a composition of the disjoint experiences $Exp^{\delta}(\delta', \epsilon_0, \zeta)$ in some entity in a proposition. Let us assume for each experience a score η represented as a tuple of satisfactory *sat* and unsatisfactory *unsat* experiences (*sat*, *unsat*) where *sat*, *unsat* ∈ [0, 1] and *sat* + *unsat* ≤ 1. As of non-additivity, a non-dogmatic experience may be expressed that is relevant in case a trustor acknowledges some deficiency in evaluating an experience. An update by new experiences merely adds to this set, $Exp^{\delta}(\epsilon_m) = Exp^{\delta}(\epsilon_{m-1}) \cup (\delta, \epsilon_m, \zeta, \eta)$ where "no experience" is denoted as (*null*, ϵ_{m}, *null*, (0, 0)), i.e. an 'empty' experience with score η = (0, 0). Dually, an experience of complete uncertainty is denoted (δ', ϵ_{m}, ζ, (0, 0)). However, before composing this set of disjoint experiences to an abstract experience, the optional decay on each experience need to be performed. Hence, the quantity of information compensates for the lack of quality [213].

Decay is an operation of forgetting / forgiveness. It is an operation that enables prompt reaction to sudden changes in behaviour by continuously adjusting the abstracted experiences. For example, when applied on time the intuition is that former experiences weigh less than recent experiences. Hence, decay is implemented as reducing the weight of a local experience (*sat*, *unsat*) with respect to its continuous datum ϵ. Central in decay is that it must not subvert the decayed experiences, but merely reduce their relative weigh, i.e. there is no evidence of the experience being less trustworthy but merely less certain. Abstractions exempt of decay are valid when assuming consistent behaviour with a goal to increase the trustor's confidence level [2] [244], making the implemented trust model's task merely to pinpoint the objective level of trustworthiness, e.g. the relation of outcomes when tossing a dice . According to Massa and Avesani [182], "most of the current research takes the assumption that every user has an objective trustworthiness value and the goal of the techniques is just to guess this correct value" [182]. However, in terms of context, such assumption is improper.

Hence, the decay is performed on each experience $Exp^{\delta}(\epsilon)$. This assures the trust property of incomplete trustworthiness. Let λ denote a decaying term by a datum ϵ, $0 \leq \lambda \leq 1$; other terms may be introduced similarly. Then a decayed experience by δ at time ϵ_n in a continuous datum ϵ is defined:

$$d_{\epsilon_n} Exp^{\delta}(\epsilon_n) = \{(\delta', \epsilon, \zeta, \lambda^{\epsilon_n - \epsilon} * \eta)\}$$

Where $\delta' \in$ *Entities*, $\zeta \in$ *Propositions*, $\epsilon \leq \epsilon_n$ and ϵ_n denotes the moment of snap-shot reducing η, i.e. increasing uncertainty. The decay factor λ defines the 'forgetting' speed where the closer to 1, the smaller the speed [256] and obviously, $\lambda = 1$ implies no decay. Hence, linearity is not required and any other means may be defined, i.e. instead of $\lambda^{\epsilon_n - \epsilon}$ for example $\lambda^{\max(1, \epsilon_n - \epsilon - 10)}$ defining that experiences within the last 10 datums are taken fully into account. The level of decay λ has also been defined as a function on forgiveness and therefore on, regret as defined by Marsh and Briggs [178].

On decayed experiences, abstraction is the means of merging them to one abstract experience at datum ϵ_n. As only abstraction on an entity δ' in a proposition ζ is reasonable, the abstraction is $Abs_{\epsilon_n}\left(d_{\epsilon_n} Exp^{\delta}(\delta', \epsilon, \zeta)\right)$:

$$Abs_{\epsilon_n}\left(d_{\epsilon_n} Exp^{\delta}(\delta', \epsilon_0, \zeta)\right) = \left(\delta', \epsilon_0, \zeta, \sum_{d_{\epsilon_n} Exp^{\delta}(\delta', \zeta)} \eta\right)$$

Hence, $Abs_{\epsilon_n}\left(d_{\epsilon_n} Exp^{\delta}(\delta', \epsilon_0, \zeta)\right)$ score is the summed score as a tuple of abstract satisfaction and unsatisfaction respectively, i.e. (*abssat*, *absunsat*) is given by $fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n} Exp^{\delta}(\delta', \epsilon_0, \zeta)\right)\right)$ where *abssat*, *absunsat* $\in \mathbb{R}$.

Not surprisingly, as $Abs_{\epsilon_n}\left(d_{\epsilon_n} Exp^{\delta}(\delta', \epsilon_n, \zeta)\right)$ denotes a tuple decayed on datum ϵ_n, the updated $Abs_{\epsilon_{n+i}}\left(d_{\epsilon_n} Exp^{\delta}(\delta', \epsilon_n, \zeta)\right)$ where $i > 0$ is a recursive function whenever the decaying factor is universal and applied on all experiences locally [53] [149] [196]. Therefore, an abstraction is a continuous function. For example, let ϵ_m=ϵ_{n+1}, then the abstract score at $\epsilon_m$is:

$Abs_{\epsilon_m}\left(d_{\epsilon_m} Exp^{\delta}(\delta', \epsilon_m, \zeta)\right)$ =

$$\left(\delta', \epsilon_0, \zeta, \ \lambda * \sum_{d_{\epsilon_n} Exp^{\delta}(\delta', \zeta)} \eta + fourth\left(Exp^{\delta}(\delta', \epsilon_m, \zeta)\right)\right)$$

Again, in case of no experience, $fourth\left(Exp^{\delta}(\delta', \epsilon_m, \zeta)\right)$ = (0, 0).

The decay serves also the purpose of giving a new chance to entities that behaved unsatisfactory. This may be implemented alike in EigenTrust [155] forcing an application to bind a newcomer with some probability, or demanding a newcomer to provide their service with minimal costs in order to gain a reputation. With decay, an untrustworthy entity will start to resemble a newcomer over time and is, hence, subject to be bound as a newcomer by a specific entity. However, as of the possible diversity in biases and performance, untrustworthiness might not be unanimous. This further argues against a global level of trustworthiness and for enabling formation of conglomerates of reciprocally trustworthy entities, i.e. a social bond.

In addition, a general model's abstract scores (*abssat*, *absunsat*) denote composed decayed experiences, i.e. a trustor's opinion in a trustee. This tuple is the opinion ω and qualifies as input for representation in a Bpdf. Hence, converting it to and from the opinion ω notation is central for the sake of calculation, as Subjective Logic functions are defined on binomial opinions. The

mapping function was originally provided by Jøsang [142] and later elaborated in [141] [148] [149]:

$$\omega \begin{cases} b = \dfrac{abssat}{abssat + absunsat + W} \\ d = \dfrac{absunsat}{abssat + absunsat + W} \\ u = \dfrac{W}{abssat + absunsat + W} \\ a = base\ rate \end{cases} \Leftrightarrow \begin{cases} abssat = \dfrac{Wb}{u} \\ absunsat = \dfrac{Wd}{u} \\ \\ a = base\ rate \end{cases} experiences$$

In this mapping function, the parameter *W* denotes the non-informative prior weight. Choosing *W* = 2 for binomial views assures initial uniform distribution of the Bpdf whenever *a* = 0.5. Higher *W* merely slows the influence of experiences down [141] [148]. More on the Bpdf and examples of these are presented in Section 4.3.3.

4.3.2.5 Calculating with Trust

Mending local abstracted experience $fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n}Exp^{\delta}(\delta', \epsilon_n, \zeta)\right)\right)$ with referrals' abstract experiences $fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n}Exp^{\delta''}(\delta', \epsilon_n, \zeta)\right)\right)$ where $\delta \neq \delta$'' demands a means to calculate with the level of trustworthiness. In line with the DSPG and (in)direct functional relations, presented in Section 4.1.4, functions for calculating the sequential ';' and parallel '⋄' combinations are demanded. These are called *discounting* and *consensus* respectively. In addition, combining several derived levels of trustworthiness on disjoint trustees is needed, e.g. an entity *Alice* may need to derive the level of trust in trustee *Bob* in proposition *x* and *Claire* in proposition *y*. This is done by a special variant of 'AND' or 'OR' as of the uncertainty, called *multiplication* and *co-multiplication* respectively. These functions are defined on opinions and have been originally proposed by Jøsang [142] and later refined in [148] [256].

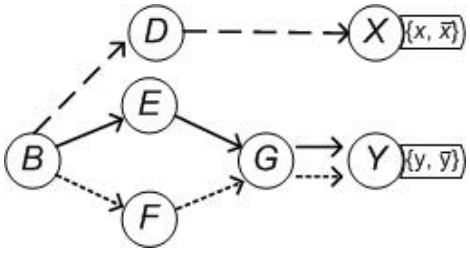

Figure 14: Two disjoint DSPG

Let us first consider multiplication and co-multiplication in Subjective logic that may be used to combine opinions of disjoint DSPGs. This may be relevant

whenever the trustor depends on several trustees to perform in a certain manner as depicted in Figure 14 where *X* and *Y* are evaluated on disjoint paths. Obviously, *B* seek to ascertain its *if*σ in a proposition on *X* and *Y* with frame of discernments $X = \{x, \bar{x}\}$ and $Y = \{y, \bar{y}\}$. The possible outcomes are therefore $X \times Y$, i.e. $\{x, \bar{x}\} \times \{y, \bar{y}\} = \{(x, y), (x, \bar{y}), (\bar{x}, y), (\bar{x}, \bar{y})\}$. Each of these outcomes need to be assigned a level of trustworthiness; where multiplication '∧' concerns the opinion in proposition $\{(x, y)\}$ and co-multiplication '∨' in proposition $\{(x, y), (x, \bar{y}), (\bar{x}, y)\}$.

To provide the functions, consider an entity *B* to have derived opinions $\omega_x^{B;D;X}$ and $\omega_y^{((B;E;G)\diamond(B;F;G));Y}$ where $x \in X$ and $y \in Y$. Multiplication on these, $\{(x, y)\}$, written $\omega_x^{B;D:X} \wedge \omega_y^{((B;E;G)\diamond(B;F;G));Y} = \left(b_{x\wedge y}, d_{x\wedge y}, u_{x\wedge y}, a_{x\wedge y}\right)$ defined:

$$\omega_{x\wedge y} = \begin{cases} b_{x\wedge y} = b_x b_y + \dfrac{(1-a_x)a_y b_x u_y + (1-a_y)a_x b_y u_x}{1-a_x a_y} \\ d_{x\wedge y} = d_x + d_y - d_x d_y \\ u_{x\wedge y} = u_x u_y + \dfrac{(1-a_y)b_x u_y + (1-a_x)b_y u_x}{1-a_x a_y} \\ a_{x\wedge y} = a_x a_y \end{cases}$$

Having the same propositions and DSPG, co-multiplication derives the outcomes $\{(x, y), (x, \bar{y}), (\bar{x}, y)\}$ written $\omega_x^{B;D;X} \vee \omega_y^{((B;E;G)\diamond(B;F;G));Y} = \left(b_{x\vee y}, d_{x\vee y}, u_{x\vee y}, a_{x\vee y}\right)$ and defined:

$$\omega_{x\vee y} = \begin{cases} b_{x\vee y} = b_x + b_y - b_x b_y \\ d_{x\vee y} = d_x d_y + \dfrac{(1-a_y)a_x d_x u_y + (1-a_x)a_y u_x d_y}{a_x + a_y - a_x a_y} \\ u_{x\vee y} = u_x u_y + \dfrac{a_y d_x u_y + a_x d_y u_x}{a_x + a_y - a_x a_y} \\ a_{x\vee y} = a_x + a_y - a_x a_y \end{cases}$$

Multiplication and co-multiplication are commutative but not distributive, i.e. $\omega_{x\wedge y} = \omega_{y\wedge x}$ but $\omega_{x\wedge(y\vee z)} \neq \omega_{x\wedge y} \vee \omega_{x\wedge z}$ and similarly for co-multiplicaiton. Multiplication is well formed except for when $a_x = 1$ and $a_y = 1$ similarly co-multiplication is well formed except for when $a_x = 0$ and $a_y = 0$ (division by zero); in this case when the opinions ω_x and ω_y may be considered as limiting values and subject to relative rates of a_x and a_y. More about these may be found in Jøsang and McAnally [151].

With respect to probabilistic calculations, calculation of belief in multiplication and disbelief in co-multiplication deviates. That is, for multiplication, the calculation of belief may appear non-standard, that in probabilistic calculations is motivated as numerical multiplication; and likewise

for disbelief in co-multiplication. The purpose of this is to get the expectation value to converge with its probabilistic peer and keeping the base rate motivated. For example, consider ω_x = (0.466, 0.074, 0.459, 0.5) with $E(\omega_x)$ = 0.696 and ω_y = (0, 0.685, 0.313, 0.5) with $E(\omega_y)$ = 0.158, for $E(\omega_{x\wedge y}) = E(\omega_x) * E(\omega_y)$, this deviation is necessary.

Having defined how to combine disjoint opinions by multiplication and co-multiplication, deriving an opinion on one proposition in a DSPG is by consensus and discounting. Consider the network on proposition provided by *Y* depicted in Figure 14, DSPG *Graph* = ({*B*, *E*, *F*, *G*, *Y*}, {(*B*, *E*), (*B*, *F*), (*E*, *G*), (*F*, *G*), (*G*, *Y*)}) where vertex *Y* has direct functional trust *df*σ on local experiences on proposition *y*; this means direct functional trust in matching proposition *y* in accordance with trust property 4 and [152]. Hence, the two paths ρ_1 and ρ_2 are:

ρ_1 = [*B*, *y*, *if*σ] = [*B*, *E*, *dr*σ] ; [*E*, *G*, *dr*σ] ; [*G*, *Y*, *dr*σ] ; [*Y*, *y*, *df*σ]
ρ_2 = [*B*, *y*, *if*σ] = [*B*, *F*, *dr*σ] ; [*F*, *G*, *dr*σ] ; [*G*, *Y*, *dr*σ] ; [*Y*, *y*, *df*σ]

To calculate the opinion from these paths, consensus and discounting are needed. Discounting, denoted ⊗, merges serialised opinions denoted ';' in the paths, i.e. $\rho_1 = \omega_y^{B;E;G;Y} = \omega_E^B \otimes \omega_G^E \otimes \omega_Y^G \otimes \omega_y^Y$. This relates, for example, *E*'s opinion in *G* by *B*'s opinion in *E*. Consensus, denoted ⊕ is the operation of combining parallel opinions denoted ⋄, i.e.

$$\omega_y^{((B;E);(E;G))\diamond((B;F);(F;G));(G;Y)} = \left((\omega_E^B \otimes \omega_G^E) \oplus (\omega_F^B \otimes \omega_G^F)\right) \otimes \omega_Y^G \otimes \omega_y^Y$$

Hence, DSGP of Figure 14 on *y*, in accordance to Section 4.1.4 is:

$\rho_1 \diamond \rho_2$ = (([*B*, *E*, *dr*σ] ; [*E*, *G*, *dr*σ]) ⋄ ([*B*, *F*, *dr*σ] ; [*F*, *G*, *dr*σ])) ; [*G*, *Y*, *dr*σ] ; [*Y*, *y*, *df*σ]

At least three different means for discounting opinions in various scenarios have been defined for arbitrary $\omega_t^{S;X}$ [154]:

$$(i)\begin{cases} b_t^{S;X} = b_X^S b_t^X \\ d_t^{S;X} = b_X^S d_t^X \\ u_t^{S;X} = d_X^S + u_X^S + b_X^S u_t^X \\ a_t^{S;X} = a_t^X \end{cases}, \quad (ii)\begin{cases} b_t^{S;X} = b_X^S b_t^X + d_X^S d_t^X \\ d_t^{S;X} = b_X^S d_t^X + d_X^S b_t^X \\ u_t^{S;X} = u_X^S + (b_X^S + d_X^S) u_t^X \\ a_t^{S;X} = a_t^X \end{cases}$$

Case (*i*) discounts the evidence while favouring uncertainty, originally published in 1997 [142]. Case (*ii*) view conflicting opinions as belief, that is, *your enemy's enemy is your friend* [144]. However, the authors note that modelling chains longer than two arcs with this methodology is doubtful. The third case (*iii*) operates on expectation values being a bad choice at high uncertainty, but might be (in special cases) the least bad choice, called base rate sensitive discounting.

$$(iii)\begin{cases} b_t^{S;X} = E(\omega_X^S) b_t^X \\ d_t^{S;X} = E(\omega_X^S) d_t^X \\ u_t^{S;X} = 1 + E(\omega_X^S) u_t^X - E(\omega_X^S) \\ a_t^{S;X} = a_t^X \end{cases}$$

In case (*iii*), expectation $E(\omega_X^S) = b_X^S + (u_X^S a_X^S)$, as before. Obviously, discounting is asymmetric, i.e. $\omega_t^{S;X} \neq \omega_t^{X;Y}$.

Contrary to discounting, consensus $\oplus$ enforces the evidence in a third party by combining parallel paths. In *Graph* consensus is needed when combining the two serial paths $\omega_Y^{(\omega_E^B \otimes \omega_G^E)}$ and $\omega_Y^{(\omega_F^B \otimes \omega_G^F)}$, i.e. $\left((\omega_E^B \otimes \omega_G^E) \oplus (\omega_F^B \otimes \omega_G^F)\right)$. The first variant of consensus was published alongside (*i*) of discounting [142] whereas only later, the consideration of a priori *a* was included [141]. For an arbitrary case $\omega_t^{X \diamond Y}$ the consensus is defined:

$$b_t^{X \diamond Y} = (b_t^X u_t^Y + b_t^Y u_t^X)/(u_t^X + u_t^Y - u_t^X u_t^Y)$$
$$d_t^{X \diamond Y} = (d_t^X u_t^Y + d_t^Y u_t^X)/(u_t^X + u_t^Y - u_t^X u_t^Y)$$
$$u_t^{X \diamond Y} = (u_t^X u_t^Y)/(u_t^X + u_t^Y - u_t^X u_t^Y)$$
$$a_t^{X \diamond Y} = \frac{a_t^Y u_t^X + a_t^X u_t^Y - (a_t^X + a_t^Y) u_t^X u_t^Y}{u_t^X + u_t^Y - 2 u_t^X u_t^Y}$$

Here division by 0, i.e. $(u_t^X + u_t^Y - u_t^X u_t^Y)$, is guaranteed as of decay λ reducing certainty. Consensus is symmetric, i.e. $\omega_t^{X \diamond Y} = \omega_t^{Y \diamond X}$.

In addition to these, conditional subjective reasoning has been defined as deduction [153] and abduction [208]. The conditional deduction and abduction on multinomial opinions have been presented in [143] [144]. For binomial opinions [153], conditional deduction and abduction is a causal probabilistic reasoning methodology that makes analysis of Bayesian networks in Subjective Logic possible [143]. However, as this thesis does not seek causal relationships on contexts, we direct interested readers to referenced literature [143] [144] [153] [208]. Moreover, trust transitivity utilising conditional deduction and abduction has been further examined by Jøsang et al. [146], in which the authors note that "despite the fact that uncertainty is taken into consideration, its value results from a sound and calculative model, rather than being an ad-hoc representation of the unpredictable nature of the transaction outcome" [146].

4.3.3 Filtering Unfair Opinions on the Bpdf

A Probability Density Function (pdf) describes the relative likelihood of a random variable to occur at a given point. The B distribution of a pdf is considered as it spans an interval [0, 1]. Hence, a Bpdf models the posterior probability. It captures uncertainty by uniformity of the distribution. Using the Bpdf for modelling trust was originally considered by Mui et al. [196] [197].

Their Bpdf did, however, not consider the 'forgetting' (decay) factor which was added by Jøsang et al. [149].

With respect to the general model, the input parameters (α, β) of a Bpdf are derived from the experiences, here (*abssat, absunsat*). This transformation is defined by Jøsang and Whitby in [148] [256]:

$\alpha = abssat + Wa$,

$\beta = absunsat + W(1 - a)$

The only input generating a uniform distribution is when α = 1 and β = 1, equal certainty in all outcomes. Hence, α = 1 and β = 1 is motivated as the initial configuration whenever *abssat* and *absunsat* = (0, 0) and base rate is 0.5; also motivating $W = 2$ as presented in Section 4.3.2.4 and yielding initially ω_x = (0, 0, 1, a). The base rate a may not be 0.5 as W may be greater than 2. Larger W merely slows the influence of evidence.

The Bpdf itself is defined by gamma functions as follows:

$$\text{Bpdf}(prob\,|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} * prob^{\alpha-1} * (1-prob)^{\beta-1}$$

A gamma function is defined for positive n as an integer as $\Gamma(n) = (n-1)!$.

The expected probability *prob* is be defined as α / (α + β). Hence, with 0 experiences and a uniform a priori expectation base rate a = 0.5 on a binomial frame of discernment with $W = 2$, α = 1 and β = 1. In Figure 15 we illustrate the Bpdf with *abssat* = 5 and *absunsat* = 1 where $W = 2$, a = 0.5 and a = 0.75. The opinions are thereof ω_x = (0.625, 0.125, 0.25, 0.5) and $\omega_{x'}$ = (0.625, 0.125, 0.25, 0.75).

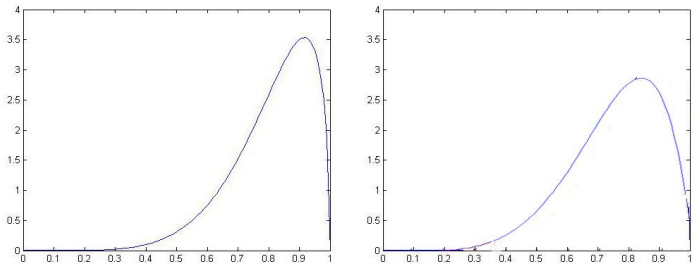

Figure 15a: Bpdf(6.5, 1.5) and 14b: Bpdf(6, 2)

The expectation value divides the signed area of a Bpdf into two equal sized halves. For the Bpdfs of Figure 15 the expectation values are $E(\omega_x)$ = 0.75 and $E(\omega_{x'})$ = 0.8125, e.g. $E(\omega_x)$ = 5/8 + 2/8 * 0,5. Additional Bpdfs are illustrated in Figure 16 where the Bpdf (3, 3) is the most uniform and Bpdf (31, 31) is the least uniform.

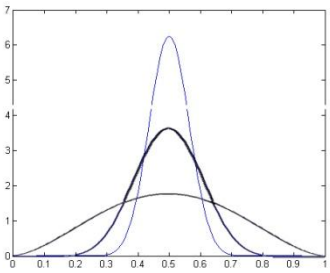

Figure 16: Bpdf(31, 31), Bpdf(11, 11), Bpdf(3, 3)

To filter unfair / biased evaluations that manifest as overly negative or positive experiences, a quantile approach has been proposed [256]. This quantile defines a lower and upper bounds as a percentage of the Bpdf within which an expectation value of any opinion considered need to fall. The quantile is defined on the Beta distribution, where a quantile q for unfair ratings is in the interval [0, 0.5] means that q percentage of the points of the $\mathrm{B}\left(fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n}Exp^{\delta}(\delta', \epsilon_0, \zeta)\right)\right)\right)$ fall under q and another q percentage over.

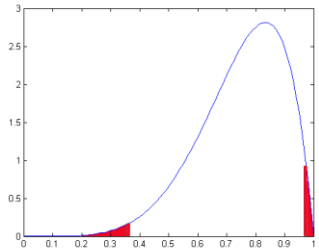

Figure 17: Bpdf(6, 2) and q = 1%

A Bpdf with q = 1% is depicted in Figure 17. Defining the lower quantile *low* = q of an opinion and *up* as the upper bound *up* = 1 – q, then for each $U \in$ *Entities* the quantile of trustor S opinion ω_{ζ}^{S} determines U's suitability as a recommender by its expectation value $E(\omega_{\zeta}^{U})$ by whether or not this is in the interval:

$$low_{\mathrm{B}\left(fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n}Exp^{S}(\delta,\epsilon_0,\zeta)\right)\right)\right)} \leq E(\omega_t^U) \leq up_{\mathrm{B}\left(fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n}Exp^{S}(\delta,\epsilon_0,\zeta)\right)\right)\right)}$$

If this predicate does not hold, then entity U's abstracted experience is not included. That is, the most unfairly positive and negative ratings are excluded with respect to the expected opinion. Noteworthy is that in contrast to this thesis and expectations on opinions, Whitby et al. [256] utilise expectation of α / (α +

β), i.e. as if discarding base rate. Moreover, they assume existence of cumulative rating vectors for each owner in the community with only one proposition. Hence, they do not consider the initial view with no experiences and they assume the existing view correct.

4.4 Success criterion

In this thesis we consider experience-based trustworthiness from the QoC perspective where trust denotes a trustor's subjective belief in a trustee to provide according to the QoC parameters it claims. Such trustworthiness may be represented as a probability or by a non-probabilistic metric. Moreover, trustworthiness should preferably model the level of (un)certainty.

Including uncertainty in experience-based trust suggests to employing the Dempster-Shafer theory. A variant of the Dempster-Shafer theory restricting the DAG of a Bayesian network to a DSPG is the Subjective Logic. The Subjective Logic framework provides a probabilistic computational model of opinions on propositions. These opinions may be mapped to and from a history of summed decayed experiences. Moreover, Subjective Logic may be used to calculate trust in context derivation as well, as each path of a (sub)polytree is trivially a DSPG. Thus, Subjective Logic provides a viable solution for the QoC parameter of trustworthiness. This leads us to state Success criterion 4:

Success criterion 4 Provide a means to compose disjoint contexts by their QoC parameter of trustworthiness and to represent the composed context.

This success criterion coins what is demanded by QoC parameter of trustworthiness in context derivation; that it affects the provided context.

With this, the need of a working incentive is emphasised. This leads us to state yet another Success criterion as an incentive ought to encourage the trustee to perform consistently and benevolently.

Success criterion 5 Defining a bidirectional incentive that encourages consistent behaviour possibly forming groups of mutually trusted entities.

The importance of such an incentive must not be belittled in an environment populated by autonomous possibly inconsistently behaving entities. Altogether, providing a scalable means for Success criterion 4 and Success criterion 5 would capture several of the central problems related to context and context-awareness.

"*Civilization advances by extending the number of important operations which we can perform without thinking about them.*" - Alfred North Whitehead

5 Trustworthy Context-Awareness

In this chapter we consider the means to capture context and its inaccuracies formally. First, we present the formal prerequisites. Second we explain how the context and context dependencies may be modelled formally. These dependencies are modelled on the user application as well as on the applications and consider a "best effort" means to capture contexts. Finally, we present how trustworthiness may be included on the context dependencies with the novel view of weighing the context with respect to its providing contexts' levels of trust.

All applications are triggered by some external means such as a key press, a remote procedure call or some other algorithmically unpredictable event. Let this (these) external means be denoted c as of context provided by the context-aware system architecture $context_S(t)$ and monitored by an application $situation_A(t)$, as discussed in Sections 2.1.3, 2.3.3 and 3.1.2. Hence, in a triggered application c is of relevance at time t, $c \subseteq S_N(t\text{-}1)$, i.e. $c \subseteq S_A(t\text{-}1) \wedge c \subseteq C_S(t\text{-}1)$. This makes $S_N(t) \neq \emptyset$ for all context-aware agents not in their initial state, i.e. all applications and user applications are fundamentally context-aware [198] [238] [275] [276]. Because of this and because the context is inherently inaccurate, the purely algorithmic model of a context-aware agent breaks down. Moreover, the inherent inaccuracy and the continuously changing qualities of contexts give rise to the QoC parameters. Of these, we consider the parameter of trustworthiness in detail.

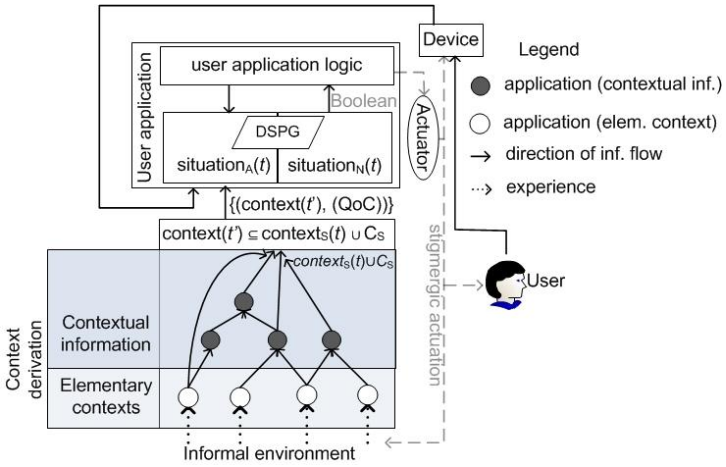

Figure 18: The context-aware processing framework

The context-aware architecture that we model in this thesis is illustrated in Figure 18. With respect to Figure 4, Figure 18 includes the notion of trustworthiness calculations through the DSPG in the user application. It models a user triggering a context-aware user application for performing a task through a device and the filing of experiences by an edge between the user and the device. These are considered contexts to the user application. The possible user inputs in addition to all other momentarily relevant contexts for a user application ($situation_A(t)$) are irreversibly abstracted as terms of a predicate. The term on trustworthiness may also incorporate user authentication policies, e.g. when we have principal $Alice \in Entities$, then $fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n}Exp^{\delta}(Alice, \epsilon_0, \zeta)\right)\right)$ with $\lambda = 0$ is trivially (0, 0). This indicates the a priori opinion ω_{Alice} = (0, 0, 1, a) with $E(\omega_{Alice})$ determined by a that may be defined by a Boolean policy, e.g. a is 0 in proposition where the policy prohibits *Alice* and 1 otherwise. Eventually, contexts in $situation_A(t)$ with affirmative predicates are candidates for being

reacted on by the context-aware user application. Of these, the ones used by the user application are captured by $situation_N(t)$.

The contexts of $situation_A(t)$ on which trustworthiness is resolved are provided by $context_S(t)$ and derived from elementary contexts in a logical topology of a polytree. In a polytree, only nodes of direct dependence need to be known as of undirected acyclicity. Each node therefore abstracts a subpolytree in which QoC parameter of trustworthiness appears as direct functional trust (no reputation or referrals) and merger may be done by multiplication and co-multiplication.

To provide the user application with a trustworthy situation, the contextual derivation needs to be analysed. We assume the algorithmic part of the applications and the user application's logic to be formally specified and their functions verified. Hence, as mentioned in Section 1.5, this thesis focuses on the formal modelling of the logical context derivation topology and on capturing the imperfection of context. Consequently, we do not use formal methodologies for analysing mathematical characteristics of a specification in a model of the implementation environment; but use the formal methodologies to model the contextual dependence and imperfection. We motivate this approach by that verification of inherently imperfect and unpredictable contexts is only possible when the imperfection of these are unjustifiably assumed [4], typically abstracted / approximated by the model [42].

For our purpose of modelling the logical topology of context derivation, use of formal deductive methods appears valid. These deductive methods "build on logical inferences and rely on theorems for proving" [91]. They rely on intermediate assertions for checking intermediate states and manage complexity in program verification with research dating back to the 1940s. On this basis, the seminal work by Hoare in 1969 [128] introduced a set of axioms and rules for correctness, called Hoare triples. Related to the Hoare triples is the *weakest precondition predicate transformer* (*wp*) semantics by Dijkstra [80] [81]. The *wp* provides an algebraic means to reason on transformation of predicates. Further, this predicate transformation semantics paved the ground for developing a formal relation of the deductive method of increasing detail in a step-wise manner from an abstract specification to a more concrete specification of the system while preserving its (mathematical) correctness. This enabled a variant of formal modelling [206] referred to as *refinement* whose mathematical foundation is based on work by Back [17] and Morgan [192]. Later, this developed into the *refinement calculus framework* [24] that relies on lattice theory. Moreover, a specification in a deductive formal methodology may include probabilities [193] [194]. These probabilities are, however, probabilities of truths that are bound by the model.

To the best of our knowledge, there are no means to measure a model's relatedness with its environment, i.e. to deduce a level of how far from the real environment the model is [4]. In addition, the model's relatedness with the environment may be subjective and when considering context, the model's

relatedness is inaccurate. To these issues, we claim the level of trustworthiness derived from an experience-based trust model valid. This is because the experience-based trust model captures a user's belief dynamically in the model and the specification's correctness, i.e. the probability of provability as opposed to probability of truth.

5.1 Formal Prerequisites

Formally specifying a system relying on a model of the environment is motivated by the desire to analyse the specified system rigorously. For analysis, a specification typically defines *what* is guaranteed, not *how*. Moreover, as the requirements specify what a system is expected to perform, a specification may be used to show the adherence with the requirements. To rigorously show this adherence, a formal specification is expressed in a language with an associated formal semantics, hereafter called formal methods.

At the moment of writing, there is an abundance of formal methods for specifying computerised systems. Roughly speaking, the formal methods can be divided into three groups: one focusing on communicational matters known as *event based* formalisms including CSP [129], CCS [189], π–calculus [190] and REO [12]; one focusing on the state of a software known as *state based* formalisms including Action Systems [19], B [5], Event B [4], Z [6], Unity [175], Hoare triples [128], *wp* predicate transformers [81]; and the *property based* formalisms (temporal logics) including LTL [207], CTL [89]. The formal method selected for specifying a specific system is often dictated by its convenience on the characterising problem setting, its user's familiarity with its semantics and its possible tool support.

For rigorous modelling of context dependencies and context-awareness where a context may trigger an actuation, the state based formalisms fit well; recall Definition 1 where a situation is modelled as part of the state. Of the state based formalisms, we use the Action System framework [19] originally developed for specifying distributed systems.

Due to its flexibility, yet formal rigor, the Action System framework is convenient for expressing novel ideas in the distributed nature of context we seek. The Action System framework also adheres to an extensive set of refinement rules [16] enabling rigorous stepwise development of a specification. Its semantics are based on the well-established weakest precondition (*wp*) predicate transformer [81], that is an alternative for Hoare logic to proving correctness [24]. Moreover, an action system $\mathcal{A}$ in the Action System framework may be part of a larger system, where the rest is modelled as the environment $\mathcal{E}$ of $\mathcal{A}$. The action systems may communicate, for instance, via global variables. This is similar to the local and distributed applications of context [48]. Drawbacks on the Action Systems framework include, from the implementer's view, the lack of tool-support and, from the context point of view, the lack of

temporalities. However, Event B [4] having tool support in the form of the Rodin-platform [92] shares many characteristics with the Action System framework.

5.1.1 Weakest Precondition Predicate Transformers of the Action System Framework

The state based methodologies focus on observing a system's *state space* and defining update statements. The weakest precondition predicate transformer *wp* is a defined based on a statement *s* and a postcondition *q*, *wp*(*s*, *q*). In this thesis we use *wp*(*s*, *q*) instead of *wp*.*s*.*q* with the wp-bracket separating the left-hand statement from the postcondition by a comma ','. This makes *wp* a function that on *s* is a predicate transformer to *q*, i.e. *wp*(*s*, *q*): $(\Sigma \rightarrow \text{Bool}) \rightarrow (\Gamma \rightarrow \text{Bool})$ where Σ and Γ denote before – after state spaces [24]. Hence, *wp*(*s*, *q*) is a composite predicate (Boolean function) identifying a set of states $\Sigma_{wp(s,q)} \subseteq \Sigma$ for which executing *s* guarantees establishing *q*, i.e. $\Gamma_q \subseteq \Gamma$.

Originally, the *wp* semantics was defined by the language of guarded commands [80] [81]. The *wp* semantics assumed that no statement may establish the *false* postcondition, i.e. that *wp*(*s*, *false*) = *False*, a property known as the 'law of the excluded miracle'. It was developed thinking of 'assigning meanings to programs' incorporating healthiness conditions in addition to being monotonic, conjunctive and continuous [22] as well as allowing nested loops etc. Of these well motivated conditions on the meanings of programs, only monotonicity has remained unquestioned when analysing the programs as idealised executable, i.e. as program specifications. The others have been sacrificed for expressivity of specification languages. For example, the continuity condition is violated by unbounded non-determinism which is a necessary property in specifications as a miraculous statement does invalidate the law of the excluded miracle. In the following, we define the semantics used in this thesis.

The predicate transformer semantics of *wp*(*s*, *q*) for any predicate *q* is defined as follows:

wp(*magic*, *q*)	= *true*	*Miraculous statement*	(1)
wp(*abort*, *q*)	= *false*	*Aborting statement*	(2)
wp(*skip*, *q*)	= *q*	*Stuttering statement*	(3)
wp(*x* ≔ *E*, *q*)	= *q*[*E*/*x*]	*Multiple assignment*	(4)
wp(*x* :∈ *S*, *q*)	= ∀*x*'. *x*' ∈ *S* ⇒ *q* [*x*'/*x*]	*Nondeterm. assignment*	(5)
wp(*sA*; *sB*, *q*)	= wp (*sA*, wp (*sB*, *q*))	*Sequential composition*	(6)
wp(*sA* [] *sB*, *q*)	= wp (*sA*, *q*) ∧ wp (*sB*, *q*)	*Nondeterministic choice*	(7)
wp([a], *q*)	= *a* ⇒ *q*	*Assumption*	(8)
wp({a}, *q*)	= *a* ∧ *q*	*Assertion*	(9)

A *wp*(*s*, *q*) is read as '*the predicate that identifies the states on Σ where executing s guarantees establishing a state that satisfies predicate q*'. The weakest precondition predicate identifying the states in which executing *magic* establishes *q* is *true*, i.e. magic applied on any state always establishes *q*. Disallowed behaviour is captured by the statement *abort* with weakest precondition predicate being *false* as *q* may never be established. Statement *skip* is a stuttering statement, not changing the state space. In multiple assignment, the variables in list *x* are assigned the corresponding expression in list *E*. Non-deterministic assignment non-deterministically assigns *x* a value in set *S* where *q* captures any value on *x* from *S*. Sequential composition of two statements *sA* and *sB* is denoted *sA*; *sB*, whereas *sA* [] *sB* here denotes demonic non-deterministic choice. This same rule for angelic non-deterministic choice would be defined wp (*sA*, *q*) ∨ wp (*sB*, *q*), i.e. the difference is that for demonic choice *q* is established by all statements (universal quantification) whereas angelic for any statement (existential quantification). For assumption statement [a], if predicate 'a' evaluates to *false*, the statement behaves miraculously whilst for assertion {a}, if predicate 'a' evaluates to *false*, the statement aborts; if the predicate evaluates to *true*, both assumption and assertion behaves as *skip*. For the miraculous behaviour, note this may never be implemented and is, therefore, not a desired statement. It is, however, necessary as a consequence of the lattice theoretical foundation of refinement calculus manifesting the common least upper bound for all elements within the lattice.

With this semantics, we say that a statement is *enabled* whenever the system is in a state where by executing the statement a state satisfying the postcondition is guaranteed. This guarantee is enforced by the guard predicate *gd* calculated with the aforementioned list of predicate transformers, defined on statement *s* as follows:

gd(*s*)	= ¬*wp*(*s*, *false*)	*Enabledness*

That is, the guard predicate identifies any state that guarantees a proper outcome. Hence, statements *abort*, *skip*, *x* ≔ *E* and {*a*} are always enabled.

With the definitions above, it is possible to define a guarded statement: [*gA*]; *sA* called an *action*. Commonly *gA* is referred to as the guard whilst *sA* as the *body* of an action.

A	= [*gA*]; *sA*	*Action / guarded statement*
gd(A)	= *gA* ∧ ¬*wp(sA, false)*	*Action enabledness*

The *wp* predicate of an action on some *q* is:

wp([*gA*]; *sA*, *q*)	= *gA* ⇒ *wp*(*sA*, *q*)	*wp of an action*

Yet, one more restriction is imposed, that actions are finitely conjunctive:

wp(*A*, *q* ∧ *r*) ⇒ *wp*(*A*, *q*) ∧ *wp*(*A*, *r*)

This implies demonic non-determinism and excludes angelic non-determinism. The conjunctivity on operators implies monotonicity, i.e.:

$(q \Rightarrow r) \Rightarrow (wp(A, q) \Rightarrow wp(A, r))$

Having defined the actions, we define repetitive construct:

$wp(\textbf{do } A \textbf{ od}, q) \quad = (\forall n.wp(A^n, gA \vee q)) \wedge (\exists n.\neg gA^n)$ *Repetitive construct*

Here $A^0 = skip$ and $A^{n+1} = A^n; A$. The repetitive construct defines that after each action some other action is enabled or the postcondition q needs to be satisfied. It also defines that the number of actions are finite. Moreover, there exist some action that establishes a state where no other action is enabled, and hence q needs to be satisfied. This state is the termination state. Termination of a construct as an obligation is known as *total correctness*; and dually, partial correctness when q is established if the construct terminate [23].

The weakest precondition predicate transformers constructs are subject to refinement ($\sqsubseteq$). Refinement is defined monotonously on predicates on the state space ordered by the relation R as a lattice [16] [22]. Hence, we say that A' refines A with the relation R as $\Rightarrow$ (logical implication) when the following condition holds:

$A \sqsubseteq_R A' \mathrel{\hat{=}} \forall q \bullet wp(A, q) \Rightarrow \text{wp}(A', q)$

As of monotonicity, A' may for a certain precondition establish a stronger postcondition q' than q guaranteed by A. That is, for a certain state, if $q' \Rightarrow q$ and $wp(A, q) \Rightarrow wp(A, q')$, then this is a refinement as well. Hence, refinement applies both to operations making the predicate transformer more deterministic as on the data structure elaborating on the process.

5.1.2 The Action System Framework and its Execution Model

The *wp* semantics and actions form the basis of the Action System framework. An action system $\mathcal{A}$ in the framework consists of an initialisation statement a_0 and a **do** … **od** repetitive construct of actions separated by nondeterministic choice []. An action system $\mathcal{A}$ is outlined as follows:

$\mathcal{A} = |[\ \textbf{var } x, y^* \bullet a_0; \textbf{ do } A_1 [] \ldots [] A_n \textbf{ od }]| : z$

In $\mathcal{A}$, x and y are variables declared by this action system. Variables x are local variables and y are exported variables, denoted by an asterisk. Statement a_0 is the initialisation statement sequentially ';' composed with a **do** … **od** repetitive construct of actions A_i. The actions within the **do** … **od** are separated by non-deterministic choice []. Variables z constitute the optional imported variables declared in the environment of $\mathcal{A}$. Hence, z and y^* form a means for communication between action systems by shared variables. All variables need to have unique names [18].

The execution model of an Action System begins with the initialisation statement a_0 assigning the variables declared by this system their initial value; if the initialisation is absent, variables are assigned an arbitrary value of their type. Initialisation is followed by the repetitive construct in which an enabled action is

non-deterministically chosen for atomic execution. This selection is demonic, hence, providing no sense of fairness. An action system terminates when no action within the **do** … **od** loop is enabled, i.e. when exiting the repetitive construct. For reactive systems abstracting variable assignments to its environment, termination is a global property and the formalism comes to show properties of execution traces. Hence, reactive systems typically show partial correctness. In addition, parallel execution of actions is possible whenever they operate on a disjoint set of variables making it equivalent to executing them in either order, detailed in [15] [18] [20] as is parallel algorithms implementable [232].

5.1.3 Action System Features

Action systems have many characterising features. Some of these originate from the flexibility of the semantics that provide a methodology in which to define theoretical features. Of these features, this section presents composition, remote procedures and prioritising; in this order.

Separate action systems may be composed in parallel, denoted ||. Consider action systems $\mathcal{A}$ and $\mathcal{B}$:

$\mathcal{A}$ = |[**var** x, y^* ● a_0; **do** A_1 [] … [] A_n **od**]|: z

$\mathcal{B}$ = |[**var** u, v^* ● b_0; **do** B_1 [] … [] B_m **od**]|: w

The composition of these is defined as follows:

$\mathcal{A}$ || $\mathcal{B}$ = |[**var** x, u, y^*, v^* ● a_0; b_0; **do** A_1 [] … [] A_n
[] B_1 [] … [] B_m **od**]|: $(z \cup w) \setminus (v \cup y)$

A composed system's variable naming remain unique for the local variables in $\mathcal{A}$ || $\mathcal{B}$ if $x \cap u = \emptyset$; and when not, mere a priori local renaming suffice. Hence, theoretically composing the environment $\mathcal{E}$ with the action system $\mathcal{A}$ at hand makes all variables local. Moreover, composition is associative and commutative as variable declaration and non-determinism have no order [18]. However, composition is irreversible and therefore, often used for analysis purposes of the whole system as a monolithic specification.

The second feature of an action system is the procedure clause, denoted '**proc**'. A procedure is a placeholder for a labelled statement that when referred to is substituted for its referral statement. Hence, a procedure may affect the enabledness of an action. In action system $\mathcal{A}$ below, p_i refers to a label and P_i to the procedure body.

$\mathcal{A}$ = |[**var** x, y^*; **proc** p_1: P_1, …, p_m^*: P_m ● a_0; **do** A_1 [] … [] A_n **od**]|: z

The procedures can, alike variables, be local or globally referable. Global procedures are denoted with an asterisk. The procedures are called by one of the three types: call-by-value, call-by-value-result or call-by-result. More on this and procedures in general can be found elsewhere [233].

The final feature is the prioritising operator in Action Systems framework. The prioritising operator gives a certain action higher priority over some other action and is denoted // [230]. It is defined on two actions *A* and *B* where *A* is prioritised over *B* as:

$A//B = A\ [] \ [\neg gA];\ B$ *Prioritising*

Hence, *B* is enabled only in states where *A* is not.

5.2 Formal Modelling of Context Dependencies

Dependency between actions in the *wp*-semantics may be expressed by sequential composition, e.g. *A*; *B*. Therefore, stating that *B* depends on *A* as *A* needs to finish before *B* in a state where *B* is enabled is valid; with the enabledness predicate $gA \wedge \neg wp(sA, \neg gB \vee wp(sB, false)$. However, with sequential composition, s*A* might enable *B*. As of this, sequential composition for expressing dependence qualifies when atomicity is guaranteed and all executing statement's behaviours are known in detail.

When modelling context, however, the assumption of atomicity is unreasonable as the contextual environment is dynamic and matters may happen concurrently. That is, having a context-aware action *A* that is to execute in a context of *B*, writing *B*; *A*; *B* unreasonably "freezes" the environment from executing. Moreover, the action resolving the context *B* is modelled to execute twice. Hence, contextual dependency needs a more flexible means to be modelled, where Boolean rigour may not be achieved, i.e. being certain that *B* is enabled once *A* finishes is impossible.

5.2.1 Situational Dependence

All context-aware agents depend on some situation captured as $situation_A(t)$. When the context deriving system provides a matching imperfect context $context_S(t)$, the situation available for providing context-awareness is captured as $situation_N(t)$. On $situation_N(t)$ a predicate is applied determining whether or not to engage in an context-aware action. However, as all contexts are imperfect, the $situation_N(t)$ is imperfect as well. Hence, a predicate on $situation_N(t)$ and its QoC parameters indicate reliance on an imperfect situation as a whole. This includes relying on the correctness of the derivation, the temporal resolution, the benevolence of the provider and many other aspects. Consequently, modelling dependence on such an imperfect matter that does not adhere to atomicity is necessarily a best effort model. Simplifying these models for formal analysis typically abstracts or assumes the imperfect matters that are causes of faults and failures [202].

A straight forward model for simplifying the imperfection alike in probabilistic analysis may not be provided for a contextual environment as of its dynamicity. This is because imperfection may be due to the implementation environment, human biases or any other inconsistent aspect. Consequently, we propose in this thesis to capture the dynamic imperfection as the QoC parameter of subjective posterior trustworthiness as an experience-based trust parameter. In a way, this extends probabilistic analysis forming the foundation for formal performance analysis and probabilistic formal methods by probabilistic choice [186] [193] considered as the QoC parameter of probability of correctness. Consequently, a predicate on $situation_{N}(t)$ should evaluate all QoC parameters where for trustworthiness, this implies a threshold on the expectation value $E(\omega) \geq z$ and / or on the opinion, e.g. $\omega_y(b) \geq 0.5 \wedge \omega_y(u) \leq 0.2$.

Consider two actions *A* and *B* of a user application and where *A* provides the context-aware functionality and *B* defines the situation by a predicate on $situation_{N}(t)$. Due to the atomic execution model of actions, what context dependence should assure is that the predicate on $situation_{N}(t)$ modelled as *gB* holds prior and after action *A*. Hence, assuring that *A* does not share variables of *gB* seems valid. However, the informal environment violates this assumption as contexts do not adhere to atomicity. Hence, a best-effort model for assuring a $situation_{N}(t)$ as *gB* is realised as an action $B = [gB]; skip$ that encapsulates a context-aware action *A*, i.e. *B*; *A*; *B*. Obviously, the body of *B* (*skip*) may be superposition refined [21] to some new functionality, e.g. filing $Exp^{\delta} = (\delta', \epsilon, \zeta, \eta)$.

As of this, the dependence operator \\ is defined in Paper IV as:

$A \backslash\backslash B = [gB]; A; B$

This operator has two important implications: Firstly it assures the context prior to engaging in executing the context-aware action and Secondly, the separation of the formal actions and contextual environment is preserved where the same context may contribute to several actions in many Action Systems. Hence, writing $A \backslash\backslash B$ assures that action *A* may not (stigmergically) enable *gB*, may only execute in context of *gB* and as *B* is executed after *A*, *A* may be guaranteed not to update the state in a manner disabling *B*. Moreover, dependence $A \backslash\backslash B$ is a refinement of *A*; *B*, i.e. $A; B \sqsubseteq_{R} A \backslash\backslash B$.

Proof. $A; B \sqsubseteq_{R} A \backslash\backslash B$

$\forall q: wp(A; B, q) \Rightarrow wp(A \backslash\backslash B, q)$

< expanding \\ >

$\forall q: wp(A; B, q) \Rightarrow wp([gB]; A; B, q)$

< assumption (8) >

$\forall q: wp(A; B, q) \Rightarrow (gB \Rightarrow wp(A; B, q))$

< definition $\Rightarrow$ >

$\forall q: wp(A; B, q) \Rightarrow (\neg[gB] \vee wp(A; B, q))$

< definition $\Rightarrow$ >

$\forall q: \neg wp(A; B, q) \vee (\neg[gB] \vee wp(A; B, q))$

< reduction of parenthesis >
$\forall q$: $\neg wp(A; B, q) \vee \neg[gB] \vee wp(A; B, q)$
< *logic* >
true □

In addition, without the atomicity assumption, A is subject to temporal granularity because terms of gB, as considered in this thesis, may change unpredictably. Strengthening gB by requiring, for example, higher expectation value is trivially a refinement with ultimate state of $E(\omega_x) \geq 1$, i.e. Boolean expectation. Abstracting context for formal analysis is easily achieved by requiring Boolean expectation and binary a priori expectation.

5.2.2 Contextual Dependencies on Disjoint Contexts

The logical topology of context derivation is, as motivated in Chapter 3, a polytree. In a polytree, the context is provided by underlying autonomous applications; called colonies [66] or situation of situations [85]. These applications process acquired context $c \in context_S(t)$ for providing another context $c' \in context_S(t)$. This process being an algorithmic behaviour may be formally modelled. Moreover, if $c = c'$, then the application performed a stuttering statement, being realistically a forwarder.

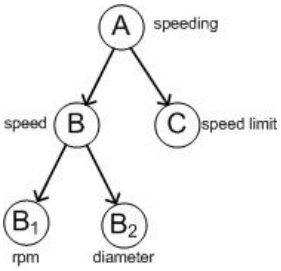

Figure 19: Contextual polytree

Assume $A\backslash\backslash B$ in a polytree with the left hand side abstracting the right, i.e. context provided by A depends on the context acquired from B. As A and B provide $context_S(t)$, they necessarily abstract the QoC parameters as well, i.e. A need to discount B's claimed level of trust $\omega_B^A \otimes \omega_\zeta^B$. Hence, $\backslash\backslash$ in calculating a level of trust is discounting $\otimes$. Moreover, if an action depends on several providers, say $B\backslash\backslash(B_1; B_2)$, the composition of the disjoint dependants trustworthiness is either multiplication or co-multiplication, i.e. $\left(\omega_{B_1}^B \otimes \omega_\zeta^{B_1}\right) \square \left(\omega_{B_2}^B \otimes \omega_{\zeta'}^{B_2}\right)$ where □ is a placeholder for multiplication or co-multiplication.

For example, consider a polytree alike that depicted in Figure 19. The context derivation of the QoC parameters for A is defined straightforward where $A\backslash\backslash(B\backslash\backslash(B_1; B_2); C)$ is a viable execution path. Other iterations on $A\backslash\backslash(B\backslash\backslash(B_2; B_1); C)$, $A\backslash\backslash(C; B\backslash\backslash(B_1; B_2))$ and $A\backslash\backslash(C; B\backslash\backslash(B_2; B_1))$ need to be viable paths as well as a consequence of contextual independence and symmetry of multiplication and co-multiplication. Hence, with respect to Figure 19, $\omega^A_{B\wedge C} = \omega^A_B \wedge \omega^A_C$ expands to $\left(\omega^A_B \otimes \left((\omega^B_{B_1} \otimes \omega^{B_1}_{rpm}) \wedge (\omega^B_{B_2} \otimes \omega^{B_2}_{dia})\right)\right) \wedge (\omega^A_C \otimes \omega^C_{spdlmt})$ that is a viable trustworthiness derivation path for *speeding* $\in$ $context_S(t)$. Notable from the trustor's point of view is that each path appears as having a length of 1, i.e. A's trustworthiness on B is determined by B's claimed trustworthiness on B_1 and B_2. Hence, B shields a colony of applications [83], here B_1 and B_2. This provides structured context derivation as stated in Success criterion 1.

Notable in a derivation such as the above is that on a principle level it differs from the means presented in Chapter 4. Here the goal is to derive a level of trustworthiness of a composite formed by a polytree, not to calculate a level of trustworthiness on each proposition by referrals in a DSPG that are eventually composed.

5.2.3 Contextual Dependencies on Similar Contexts

In the polytree, several agents may provide the same context. In this case, an abstracting agent composes these readings. This is realistically the case for triple modular redundancy or when combining the readings from a new sensor and an old with greater certainty but less belief due to wear and tear.

Existing work addressing this problem include Grossman et al. [113] who addressed the readings' inconsistencies as the arithmetic mean of the smallest and largest distance on equally distributed situations. They also considered uncertainty as the spread of the readings modelled by a restricted probability density function and trust simplified to belief. They do consider a binomial approach on providers' reliability and have all the metrics with a focus on deriving probabilities on a proposition, e.g. what is the probability of reading *a* being closer to *x* than *b*. Another view considering calculations of confidence in a situation by weights of the context with respect to their confidences is presented by McKeever et al. [219].

Our view is different as we consider trustworthiness in a well-defined proposition and include calculation of trust and merger of context, i.e. as if knowing a context reading with certain doubts on it as opposed to inconsistent and uncertain readings and seeking to calculate a merged proposition's level of trust. To the best of our knowledge, this is a novel view. For brevity, we assume equal base rates 0.5 and consider only the QoC parameter of trustworthiness. The proposed merged c' $\in$ $context_S(t)$ by an agent S is then the weighted average of the context readings c_i with respect to the expectation value of the provider X_i

$\subseteq$ $context_S(t) \setminus c$' on a proposition ζ and its corresponding level of positive experiences *abssat*. Hence, we define the merged context c' as follows:

$$c' = \frac{\sum_i^n (abssat_i * E\left((\omega_{X_i}^S \otimes \omega_\zeta^{X_i})\right) * c_i)}{\sum_i^n (abssat_i * E\left(\omega_{X_i}^S \otimes \omega_\zeta^{X_i}\right)}$$

For example, with two opinions $(\omega_{X_1}^S \otimes \omega_\zeta^{X_1})$ = (0.3, 0.5, 0.2, 0.5) and $(\omega_{X_2}^S \otimes \omega_\zeta^{X_2})$ = (0.16, 0.8, 0.04, 0.5), the corresponding *abssat* and *absunsat* are (3, 5) and (8, 40) respectively with W = 2 and $E(\omega_{X_1}^S \otimes \omega_\zeta^{X_1})$ = 0.4 and $E(\omega_{X_2}^S \otimes \omega_\zeta^{X_2})$ = 0.18. Let the provided contexts of X_1 and X_2 be c_1 = 20 and c_2 = 10 respectively, then the weighted average is 14,545 indicating that c_2 had slightly greater influence because greater certainty in its provider X_2 though less trustworthy. As c' base on two disjoint readings, composing their QoC is necessary as well. For trustworthiness, the outcome is similar to that of adding the abstracted scores of the opinions, i.e. $fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n} Exp^S(X_1, \epsilon_0, \zeta)\right)\right) + fourth\left(Abs_{\epsilon_n}\left(d_{\epsilon_n} Exp^S(X_2, \epsilon_0, \zeta)\right)\right)$, in this case (11, 45). This turns out to be the same as consensus $\oplus$, i.e. $(\omega_{X_1}^S \otimes \omega_\zeta^{X_1}) \oplus (\omega_{X_2}^S \otimes \omega_\zeta^{X_2})$ = (0.1896, 0.7758, 0.0344, 0.5). Consequently, weighted average of context with respect to its abstracted satisfactory experiences and expectation value seems to be a viable solution for enforcing the certainty in the weighted context.

"*A computer shares with mathematics the property of being at the same time the queen of science and technology and the most humble servant*." – Heinz Zemanek 1980

6 Description of Papers

In this chapter we briefly present the author's scientific publications that relate to context, context-awareness and trustworthiness. Each publication is described separately with an analysis on its contribution with respect to the stated success criterions. Reprints of these publications are available in Part II of this thesis.

Having presented context, context-awareness, trustworthiness and how these concepts fit together in a formal framework; the author's scientific publications may be considered as milestones on this track. The publications are reprinted in Part II. Each of them considers a particular aspect from a certain point of view. This section lists the main contribution of each paper, how it fits the research, what challenges and success criterions it addresses and how. Moreover, the author's role in each of them is described. The papers are presented in a chronological order.

Paper I. An Abstract Model for Incentive-Enhanced Trust in P2P Networks

Mats Neovius, "*An Abstract Model for Incentive-Enhanced Trust in P2P Networks*". In: Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuanshun Dai, Laurence T. Yang (Eds.), *Embedded and Ubiquitous Computing - EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES, Nagasaki, Japan, December 6-9, 2005.* , Lecture Notes in Computer Science 3823, 602 - 611, Springer Berlin / Heidelberg, 2005.

This paper presents a model for facilitating benevolent behaviour on a uniform event by an incentive in a Peer-to-Peer (P2P) network. The P2P network is organised by two interconnected distributed hash tables, one for the long term ID and one for the session ID. It defines a P2P system in a manner requiring all experiences to be stored on the 'live' entities, i.e. by the entities having a session ID. A recovery method by a logical expression for suddenly dropped entities is also outlined.

The level of benevolence is derived from experiences (feedbacks) as experience-based trust. The paper also presents how to propagate, distribute and compose experiences in a decentralised P2P network by means of Subjective Logic. It implements decay with two P2P specific operators: by time and sociality. Sociality is motivated by considering the information rightfully from the "long tail", i.e. in a setting fitting the P2P environment. Moreover, the paper considers the base rate for acquiring the expectation value as the expectation value of the general opinion, i.e. providing the ability to trust despite bad reputation [150]. It also addresses whitewashing by assigning a newcomer minimal privileges. In addition, as each entity utilises a set of trustworthy entities for derivation, the incentive for any entity to behave consistently is its influence on entities trusting it.

This paper addresses Success criterion 5 by valuing consistent behaviour. In terms of context, it considers one context on which all experiences are expressed. The author is the sole author of this paper.

Paper II. A Design Framework for Wireless Sensor Networks

Mats Neovius, Lu Yan, "*A Design Framework for Wireless Sensor Networks*". In: Khaldoun Al Agha (Ed.), *Ad-Hoc Networking: IFIP 19th World Computer Congress, TC-6, IFIP Interactive Conference on Ad-Hoc Networking, August 20-25, 2006, Santiago, Chile* , IFIP International Federation for Information Processing 212, 119 - 127, Springer, 2006.

This paper presents a general architecture of a wireless sensor networks (WSN). The amorphous WSN is modelled in a 3-dimensional architecture with communicational layers, vocational segments and management planes. The communicational layers refer to the level of abstraction of data whilst the vocational segments to a node's capabilities. The layers and segments are called vertical and horizontal reasoning by Broens [48]. The sensor network specific management planes of power-, mobility- and task planes are implemented on each node [9]; the 3rd dimension. On such a framework, the paper stresses that the main load is on the diagonal ellipse. Thus, each component being a part of the derivation chain performs some functionality with a role in the system that may be illustrated by its location within the ellipse.

This paper introduces a framework for developing applications relying on a decentralised network populated by autonomous agents demanding collaboration in order to deliver for some inquiry. It motivates and briefly addresses Success criterion 3. The WSN was chosen as it relates to context (this paper was written simultaneously with paper III) and the sensor motes were easily acquirable. Moreover, the spirit in which the paper is written supports the idea of a logical topology of a polytree.

The author's contribution to this paper was approximately 85% of the work. The co-authors mainly contributed in the section on middleware and in discussing the coining of the framework.

Paper III. A Formal Model of Context-Awareness and Context-Dependency

Mats Neovius, Kaisa Sere, Lu Yan, Manoranjan Satpathy, "*A Formal Model of Context-Awareness and Context-Dependency*". In: Van Hung Dang, Pandya Paritosh (Eds.), *Proceedings of the fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06), 2006.*, 177 - 185, IEEE Computer Society Press, 2006.

This paper formally considers how context is derived and how a context may be modelled in a context dependent entity. It treats the domain outlined in Paper II with respect to specifications of nodes in a context-aware scenario. It is inspired

by the WSN research as well as the co-authors' previous publications on context in mobile computing [267].

In this paper, context is considered uncontrollable and defined as "a setting in which an event occurs". These contexts are rudimentary evaluated by a context dependent entity as terms of a predicate, called context guard. Whenever context guards are false, the paper models some other action enabled when out of context. This is fundamental for the sake of showing the termination condition in refinement. The paper also shows a strategy to refine context with respect to its definition on context. In addition, the paper outlines how context is derived in line with Paper II and how a provider abstracts its underlying architecture from its consumer.

The paper contributes to Success criterion 1, Success criterion 2 and Success criterion 3 providing a hierarchy and treating the context merely by a predicate. The author's contribution to this paper was approximately 35% of the work with the main contribution in setting the idea of treating context merely as a general uncontrollable variable that may change unexpectedly.

Paper IV. Formal Modular Modelling of Context-Awareness

Mats Neovius, Kaisa Sere, "*Formal Modular Modelling of Context-Awareness*". In: Frank S. de Boer, Marcello M. Bonsangue, Eric Madelain (Eds.), *Formal Methods for Components and Objects, 7th International Symposium, FMCO 2008, Revised Lectures*, 102-118, Lecture Notes in Computer Science vol. 5751, 2008.

This paper presents how a context-aware application may integrate a context (situation). It builds on Paper III as deriving context. The idea rose from considering combining paper II and III more rigorously, capturing the difference between a context and a state, defining a means to treat the inherently imperfect contexts formally. The paper considers context pragmatically as exclusively updated globally readable variables. Hence, the contexts are considered as *read-only* variables that only the application that *publishes* that context may update.

The main contribution of the paper is the context dependence operator \\, as presented in Section 5.2.1. The \\-operator formally defines how context(s) may be utilised by a context consumer. As the contexts are inherently imperfect and inconsistent, \\ provides a best effort model. This dependence operator extends mapping context as a mere term in the guard predicate by assuring the contextual condition *gB* to hold (as rigorously as possible) throughout execution of context-aware consumer functionality *A*.

The view conforms to the ellipse of paper II. It addresses Success criterion 1, Success criterion 2, Success criterion 3 and Success criterion 4. The author's contribution was approximately 65%.

Paper V. Mastering the Relevance of Subjective Information in Ubiquitous Computing

Mats Neovius and Kaisa Sere. "*Mastering the Relevance of Subjective Information in Ubiquitous Computing*". Submitted to International Journal of Networked Computing and Advanced Information Management (IJNCM) Special issue on Social Informatics and COMputing (SICOM).

This paper presents how sets of entities may be abstracted as a group. Such a group is bonded by likes on some proposition and behaves as an entity in its own right. It abstracts subjective experiences of entities being members of this group. This bonding also defines the specificity in a proposition. Hence, a group is a virtual entity that provides a composed referral opinion and in the context of this thesis, eases the user application's derivation of trustworthiness. The paper illustrates this by an example. The paper considers, for brevity, a very simple grouping. However, quantiles or more advanced grouping capturing overly positive and negative ratings could easily be defined. The views presented are, to the best of our knowledge, novel in terms of approach as well as proposed solution.

The paper addresses Success criterion 4 by composing experiences as a group. Moreover, it addresses Success criterion 5 by strengthening a bidirectional incentive for the members of a group to be consistent in order to acquire more influence. The author's contribution was approximately 90%.

"*The most important step in getting a job done is the recognition of the problem. Once I recognize a problem I usually can think of someone who can work it out better than I could*." – Leo Szilard 1961

7 Discussion and Achieved Results

In this chapter we discuss and analyse the achieved results in a systematic manner. We discuss the role of trustworthiness on context derivation, trustworthiness on a situation, incentive for consistent behaviour as well as means to formally model this. Each section shares the following structure: presenting the problem, contribution of this thesis, motivation, impact and objections against the presented approach. The goal is to convince the reader of the validity of our approach.

Trust and context are related by, among others, the QoC parameter of trustworthiness that captures a context consumer's uncertainties related to the provided context. To model this relation in a structured manner, we consider these aspects from a formal point of view. However, rather than striving to verify mathematical characteristics on such a relation of inaccuracy, this thesis utilises formal methods merely as a means of expression. Hence, the focus in this chapter is on imposing a critical discussion on the view taken with respect to related work. This chapter further stresses the contribution of this thesis by elaborating on how the contribution fits the stated success criterions. Table 1 provide the reader with a view of which paper or section addresses what challenge and success criterion.

Table 1: Challenges, success criterions and contribution

Challenge	Success criterion	Papers	Other
Challenge 1	Success criterion 1	I, V	
Challenge 2	Success criterion 2	II, III, IV	
Challenge 3	Success criterion 3	II, III, IV	Polytree Section 3.5
Challenge 4			
Challenge 5	Success criterion 4	I, V	Polytree Section 3.5, Section 5.2.3
	Success criterion 5	I, V	

This chapter is divided in four subsections Section 7.1 discusses the role of trustworthiness on contexts in derivation whereas Section 7.2 the trustworthiness perceived on a situation. Section 7.3 discusses the incentive to behave trustworthy and Section 7.4 brings forward the formal modelling of this. All subsections share the same structure. They present in this order the problem (i), the contribution (ii), the motivation (iii), the impact (iv), and (some) objections (v). The problem (i) considers difficulties raised and acknowledged by existing work with fitting the setting of context and context-awareness. The contribution (ii) is presented with respect to the stated success criterions followed by motivation (iii) explaining the made design decisions. This is followed by impacts (iv) the contribution may have in the field of study and finally, a critical view (v) on the results presented is taken, to answer points of criticism and objections.

7.1 Trustworthiness of Context

A context's QoC parameter of trustworthiness models the level of probability of provability of this context as claimed by its provider. Sometimes this level is considered stable and modelled as a term of a predicate abstracting the context and its QoC parameter of trustworthiness. Abstractions alike are motivated only

in very specific settings where the context's imperfection is minimal, e.g. the prominent implementation of the B method modelling the automated Paris metro line nr. 14, as this metro line have, among others, platform edge doors at all stations. In such environments it may be justified to model context and ignore its imperfection providing the rigorous foundation from which formal methods in software development derive their strength [170].

Broadening the domain of discourse to open networks, such as the ubiquitous computing environment, the need for adaptive QoC parameters capturing the changing environment is evident. This characteristic of context is commonly acknowledged as the inherent imperfection [117] [121] [205]. Moreover, this imperfection breaks down the purely algorithmic model [235], excluding the otherwise very interesting situation lattices [268] [269]. Thus, a ubiquitous user application is indeed merely a piece of technology that provides the user a means to perform a task [26] without considering the environment. Consequently, for considering context formally, the context(s) imperfection needs to be encapsulated and provided separately, for example, in the QoC parameters. Of the QoC parameters, the parameter of trustworthiness is considered in detail in this thesis.

This thesis contribution is in defining and providing a means to capture and calculate the QoC parameter of trustworthiness, based on experiences. Moreover, a novel means to increase certainty by disjoint contexts on the similar contexts by weighing them by their trustworthiness is provided in Section 5.2.3, hence addressing Success criterion 4. Consequently, this thesis provides a comprehensive view on how a context is derived from a set of elementary contexts experiences and how the QoC parameter of trustworthiness emerges by combining the merged discounted sources by multiplication and co-multiplication. This context derivation is modelled as a polytree in Section 3.5, that is a contribution of this thesis. Hence it addresses Success criterion 3. Moreover, modelling QoC parameter of trustworthiness on a well-defined proposition, trustworthiness may be used for suitability purposes stated in Success criterion 1.

This thesis has motivated the QoC parameter of trustworthiness to be experience-based. Moreover, as trustworthiness builds up and changes, distinguishing between 'don't know' and 'equally trustworthy as untrustworthy' is fundamental. Hence, the QoC parameter of trustworthiness may reasonably be based on Dempster-Shafer theory for capturing the level of uncertainty, as it is a probabilistic matter and is derived from the acquired experiences. In addition, as trustworthiness is applied on inherently imperfect context, no fixed level of trustworthiness may be assumed. As of this, the Subjective Logic presented in Section 4.1.3 and Section 4.3.2 is selected as the computational model. The Subjective Logic framework provides a logic for calculating trustworthiness in a DSPG on an arc of a polytree.

A possible impact of these findings is to encapsulate the QoC parameter of trustworthiness from the context. This makes the algorithmic part of context

processing subject to formal reasoning. Moreover, as providing a means to merge disjoint contexts of a polytree, trustworthiness in a truly hierarchical topology is possible. Therefore, the applications providing context, as presented in Sections 2.1.4 and 2.3.1 may be considered hierarchically supporting the hierarchical context models of Section 3.3.2.

These findings may be objected to by that reading a value, as is the case for elementary contexts captured by a sensor, trust on it is trivial and subject to probability of correctness. Such probability of correctness would be subject to probabilistic analysis within a formal model, more on this in Section 7.4. However, trustworthiness as presented in this thesis is an artefact capturing the informal view and is based on insufficient evidence and is, therefore, very apt to context. For a specific elementary context, the inherent imperfection has been empirical found in a survey from industry [137] to relate to mechanical wear, dirt, human errors and environments affects, among other forms of informal imperfection. Obviously, this is a motivation for the existence and definition of the QoC parameter of trustworthiness [54]. Further, the Subjective Logic proposed in this thesis to be used to calculate with these insufficiencies can also handle absolute levels of trust (dis)belief ($b \in \{0, 1\}$) of non-aging ($\lambda = 1$) (in)correctness, making Subjective Logic behave like Boolean 'AND' and 'OR'. Such Boolean certainty on any non-algorithmic matter is, however, unreasonable. Research has pointed out that time takes its toll on even the seemingly permanent elements, such as DRAM [229]. In addition, even an automatic theorem prover is subject to trust in terms of trusting the author of the prover to have implemented the inference rules properly [198]. Together, these add to the need of an adaptive, incomplete quality parameter, i.e. to the need of trustworthiness.

Objections on the logical topology of deriving context include the motivation of relaxing the polytree structure to a directed tree where each context may contribute at most once per matter. This objection is motivated, however requiring a universal ontology for defining the 'once per matter', which does not exist. This becomes prevalent in hierarchical structures, e.g. calculating average speed requires a reading to be included at most once. Hence, we consider a polytree as a reasonable logical topology for deriving context with trustworthiness.

Objections on the means to derive the experiences demanded for calculating experience-based trust may be criticised as compromising the idea of context-awareness due to requiring human interaction [90]. This point of criticism is very valid. However, this thesis takes no stance on whether or not these are automatically or interactively provided, e.g. by triple modular redundancy or by the cognitive entity.

7.2 Trustworthiness on Situations

A situation abstracts a set of contexts in the logical topology of a polytree. It shares all aspects of a context, including the QoC parameters providing a partial view of the informal environment for a user application. The user application considers a situation by a predicate where trusting a situation is equivalent to depending on it to represent a set of contexts correctly. In this setting, the user application may affect a situation only through actuators that may stigmergically affect the contexts.

The trustworthiness a user application perceives in a situation is similar to that used for deriving the situation except for the logical derivation topology being a DSPG, i.e. referrals are included to recommend the situation. This provides the user application with means to calculate an accurate and timely level of trustworthiness. However, to the best of our knowledge, regardless the abundance of research on computing with experience-based trust and quite a few proposals outlining the QoC metrics, research proposing a usage for the QoC parameter of trustworthiness has not been considered.

The contribution of this thesis consists in defining a mapping from the uncertain situation to the formal context consumer by a predicate on some threshold on context terms. The view accepts the fact that this is uncertain and the outcome may, therefore, be undesired. The mapping does thereby define the cooperation threshold weighing risk and profit [178] where a discounted opinion may be ascertained by referrals experiences in a logical topology of a DSPG. Moreover, papers III and IV contribute to the view of how context may be introduced formally to a user application. These papers support separation of concerns between context derivation and usage. They also consider a situation effectively as terms of a guard. Hence, these papers and Part I of this thesis collectively addresses Success criterion 2.

The motivation of this approach relates to the separation of concern and abstraction of details where a user application yearns for a means to map the informal environment to a formal environment. For this, the inherent inaccuracy of context must be captured and eventually abstracted to a Boolean at point of actuation. This is the effect of the context guard of the \\-operator that additionally demands atomic behaviour of the environment.

The impact of abstracting a situation as terms of a predicate is that an enabled action may be provided in a certain situation characterised by the quality. Optimally, the situation guard's cooperation threshold is restricted to what would otherwise be assumed correct in systems and forms therefore, a mere means to verify the assumptions on contexts. Obviously, as a predicate is a Boolean valued function, this evaluation is irreversible and enables thereafter formal reasoning on the construct.

Objections regarding the taken view include questioning the use of formal modelling on a situation derived from inherently inaccurate sources. This

objection is very motivated. It is however discarded in this thesis, because all adaptive functionality including all applications of practical relevance is context-aware and all contexts are inherently inaccurate. Hence, a formal view on any non-mathematical domain of discourse assumes and accepts this inaccuracy of the model if it provides a Boolean argument. Therefore, employing trustworthiness as presented in this thesis is advantageous because it provides a sound and realistic means to model this inaccuracy adaptively. Moreover, the Subjective Logic with absolute opinions converges to Binary 'AND' and 'OR' [144] if necessary. Hence, using Subjective Logic on absolute contexts will not divert the formal model motivating the claim that the presented means extend the traditional means of formalising matters. In addition, Subjective Logic on dogmatic opinions (no uncertainty) converges with classical probabilistic systems [143]. Hence, Subjective Logic also extends on the traditional Bayesian analysis.

Another argument against the presented approach is the use of experience-based trustworthiness that eventually is abstracted by a policy as terms of the situation guard. This objection is discarded in this thesis by motivating experience-based trust instead of policies by the need to adapt to changes. Experience-based trust can further be argued against with the motivation that people perceive trustworthiness in people, not technology [99]. As technology is fundamentally manmade and run on an infrastructure that always have a human stakeholder, relying on the engineers' work and the stakeholder supporting this system surely is subject to trustworthiness. In addition, questioning how experiences forming the opinions of a DSPG for deriving trustworthiness in a situation are acquired is motivated. This is indeed a fundamental issue that in related work is often discarded and experiences are merely assumed to exist and be correct. We consider this to be provided by a cognitive user.

7.3 An Incentive for Behaving Trustworthy

In real life, the incentive for behaving according to some scheme is typically money, fame or some other craved benefit or contrary, fear of sanction in terms of fining, reduced reputation or something alike. All of these incentives rely on the identification of the counterpart and common basic desires, e.g. freedom, fame and wealth. The levels of these desires are enforced by the masses as what is considered socially appropriate or by third parties enforcing laws such as the police and court. As no entity has the role of the real life third party entities in computerised communication, collaboration relies on mutual trust between entities that share appreciation, i.e. views on appropriateness and bias. Hybrid incentives may be present in case of credentials enforced by real-life contracts, e.g. the university network is only available to users who have signed the terms of use, making the user subject to real-life laws. Nevertheless, incentives are as

central in computerised interaction as in real life. Live implementations on the open computerised network populated by egocentrically behaving entities include Ebay.com's reputation score, Slashdot's karma and Google's page rank.

To this, this thesis contributes by outlining an implicit incentive in a setting of context where the incentive for providing and acquiring a highly trustworthy context is bidirectional. This incentive is based on trustworthiness. Providing highly trustworthy context assigns the provider higher influence on the context consumer's decision and an increased possibility to exceed the threshold of being bound. Dually, an incentive for the context consumer to bind high quality context(s) is reduced computational load. Hence, the contribution addresses Success criterion 5 encouraging consistent and benevolent behaviour.

The approach is motivated by the sheer necessity of providing an incentive. This incentive is defined and enforced by conglomerates of mutually trusted entities, i.e. by the group an entity is associated with.

The impact is that the set of entities an entity identifies with share to a high degree the biases and appreciation by a proposition, e.g. the set of entities sharing the perception on proposition *cold drink* may be different from that in proposition *cold climate*. Characterising for such entities of a set of mutually trustworthy entities may be that when they interact, they are likely to assign satisfactory experiences to each other. Hence, such a set of entities forms a code of their own with respect to expectations, an approach further discussed in Paper V.

Objections against an incentive would typically relate to its computational costs, difficulties in distribution and it attracting fraud. However, in an environment populated by autonomous entities, the incentive is necessary. The computational costs are when implementing Subjective Logic reasonable compared to other options, e.g. matrix multiplication as in EigenTrust. Moreover, the experiences are distributed upon request and fraud may collaboratively be noticed and reacted to. Other related means to provide an incentive for benevolent behaviour include negotiations revealing increasingly sensitive data and therefore, tying entities increasingly to each other. However, we consider incentives as presented in the thesis to include this aspect as more trustworthy providers have greater stakes than less trustworthy ones in the event of an unsatisfactory experience.

Objections may also relate to how the appreciation is evaluated and distributed if several situations are used. This point of criticism is valid. Solutions may relate to distributing this according to the weight, importance or any other means. However, as noted in Section 1.5, this thesis does not consider distribution of appreciation.

7.4 The Formal View on Contextual Dependency

The importance of structured reasoning is augmented in complex systems such as ubiquitous systems. Often complexity is addressed by decomposing a system to manageable parts. Context makes no exception in this sense. However, as context is inherently imperfect, a formal specification of a context-aware agent is relative to its model's relatedness with the reality, behavioural assumptions and restrictions. Hence, formal analysis is possible only given irreversible mapping of the informal environment to a Boolean, i.e. approximating and assuming characteristics on what is modelled. However in deriving a situation, the imperfection of contexts this situation depends on needs to be considered. In related research on formal methods and their applications, this mapping is typically evaded by underlining the model as an abstracting entity on an approximated reality, i.e. Boolean assumptions are made on the elementary contexts. Moreover, this model is frequently considered to encompass all of the relevant aspects that are being specified. However, "should this approximation be too far from the real environment, then it would be possible that our software would fail under unforeseen external circumstances" [4].

The contribution of this thesis is in line with Papers III and IV; an application conducts actions only in some context. This is modelled by the predicate that defines a threshold on the context that in Paper III this is called the context guard and in Paper IV is modelled by the predicate that needs to hold before and after the application with the \\-operator. On these predicates, including the QoC parameter(s) is straight forward. On this matter, Paper V and this thesis' part I provide insight. Hence, Success criterion 5 is addressed.

The motivation for expressing dependence on an imperfect context in the first place is simply that an elementary context captures the informal world and must, therefore, not be assumed formally. Moreover, as the context may be subjective, a formal interpretation in terms of a model is void. Hence, subjectivity needs to be captured by some of the context's quality parameters. In this thesis this parameter is the trustworthiness QoC parameter that bases on user application specific experiences on the provider as a whole.

Objection on the use of formal methods on inherently inaccurate context are many. Mainly these relate to the fundamental differences among these. In the following the most prominent from this thesis point of view are outlined.

Criticism on using experiences instead of well-founded probabilistic systems and their implementations on formal methods is evaded by the different views taken. This thesis considers trustworthiness to be dynamic, subjective and to build up from initial uncertainty, motivating Dempster-Shafer theory over statically provided probabilities [193] [194].

Valid critic regarding the dependency operator \\ is that it in fact coincide with sequential composition ';'. This holds true when atomicity is assumed and

context is provided as *read_only*. Dually however, as $A \backslash\backslash B$ is a refinement of $A;B$, i.e. $A;B \sqsubseteq A \backslash\backslash B$, and the meanings coincide, expressing dependency by \\ assuming atomicity and *read_only* is valid. However, when considering context capturing the informal environment and assuming no atomicity, the algorithmic model breaks down. For this, \\ provides a supreme model over ; expressing dependence whose realistic implementation depends on the left hand side action's temporal granularity with respect to the contexts' temporal resolution.

This temporal granularity brings up the next point of criticism, that context breaks the atomicity of an action. This is the case per definition of context that is inaccurate and unpredictable in the sense that it is not created by a computer, i.e. in $A \backslash\backslash B$ the parameters of gB may change during execution of A. Assuming the algorithmic part of an action to adhere to atomicity and modelling actuator *Act* as a separate action that A enables, *Act* may not be triggered before B has executed. Hence, a designer needs to decide whether to accept uncertainty or to assume unjustifiably context updating to be atomic. Moreover, questioning for $A \backslash\backslash B$ whether A depends on B or vice versa may arise. This is only motivated when context is interpreted as terms of a predicate incapable of disabling itself and when context adheres to atomicity. If this was the case, a context's temporal resolution is lost.

Other criticism includes that of using formal methods in the first place on a non-formal matter like context. This point of criticism is very valid as the context compromise the means of mathematical analysis which is the catalyst for formalising in the first place. It also scales to the fundamental difference between mathematical modelling and engineering. On this stance, this thesis lies in between as experience-based trust could be considered a kind of testing. However, in this thesis the aim is not to prove mathematical characteristics but to use the formal methods for providing a means to model and reason on context in a structured manner. Again, Subjective Logic behaves like Boolean logic when assuming atomicity and an absolute level of trust. Hence, replacing axioms with the QoC parameter of trustworthiness is an improvement. Moreover, dependency \\ executes equivalently to ; in case of atomicity and independence. Consequently, all operators and means fall back on their traditional use, further highlighting the contributions validity and implementability.

"*Experience seems to most of us to lead to conclusions, but empiricism has sworn never to draw them.*" - George Santayana

8 Conclusions and Future Perspectives

In this chapter we summarise the thesis in terms of contributions. We also consider some future perspectives of context and context-awareness in ubiquitous computing and the role of trustworthiness in this.

This thesis considers a means to formally model contextual dependencies on inherently inaccurate contexts derived from a ubiquitous computing architecture. The Action System framework featuring a means for structured correct-by-construction (refinement) is used as a formal framework for logic-based context modelling and analysis. As the Action System framework is based on a well-established mathematical-logical theory, the challenge relates to modelling a context's inherent imperfection. This contextual imperfection is captured by the context's QoC parameters that are modelled as a context's metadata. Of these QoC parameters, the parameter of trustworthiness, noted as a challenging parameter by related work [54] [236], is examined in this thesis in greater detail. Moreover, as context is derived, challenges with respect to propagating the QoC parameter of trustworthiness are considered. Hence, in an idealised system a context with QoC parameters would provide the user application a view of the environment that is more realistic than if assumptions / approximations on the environment would have been modelled.

In the ubiquitous computing architecture that is populated by autonomous agents, a context consuming agent is in this thesis considered to acquire contexts provided by other autonomous agents. As this context consuming agent may not assume or enforce any conditions on the context provided, the QoC parameter of trustworthiness evaluated by the consumer on the provider is essential. This trustworthiness is considered to be based on subjective experiences. It captures a level of unwarranted reliance the context consumer perceives on a provider momentarily in a specific proposition. Hence, trustworthiness is considered experience-based and its level is non-monotonic. Moreover, as initially there are no experiences, the QoC parameter of trustworthiness needs to include a factor of uncertainty as opposed to certainty. Thus, trustworthiness is not a probability of truth captured by the QoC parameter of probability of correctness adhering to additivity, but a probability of provability referring to concepts as 'belief', 'doubt', 'evidence', 'support' [204]. This motivates Dempster-Shafer theory as a candidate for representing trustworthiness. Moreover, as an experience is entity- and proposition-specific, it is subjective. In addition, this thesis provides a general model for managing this history including abstraction of it for the sake of a referral's ability to preserve privacy, i.e. supports reputation-based trustworthiness

To effectively address these aspects, this thesis considers the Subjective logic framework. Subjective logic is experience-based addressing uncertainty where the level of trustworthiness is considered as functions on constructs called opinions. It is a probabilistic logic related to Dempster-Shafer theory being a generalisation of binary logic and classical probabilistic logic [143] [144]. It is also related to Bpdf by unique bidirectional transformation rules. The probability density function with a B-distribution (Bpdf) is represented as a tuple (α, β) and fits the abstracted experience tuple (x, y) where x, y denote the level of subjective satisfaction and dissatisfaction. The abstraction (aggregation) of such subjective experiences is considered as simple summation of the decayed

experiences. Each experience is represented by a tuple (x_i, y_i) for $i = 1, \ldots, n$ and $x_i + y_i \leq 1$ with $1 - x - y$ denoting the level of uncertainty, on which decay by a continuous datum reduces the certainty. Hence, decay reduces the weight of evidence of satisfaction and dissatisfaction of an experience and captures the fundamental assumption of context being ever changing, always incomplete and non-monotone. For the sake of decision, a posterior *expectation value* adhering to additivity may be derived assuming a provided a prior base-rate, denoted *a*.

In modelling this formally in the Action System framework, the inherent imperfection of context motivating trustworthiness as a QoC parameter raises some concerns. These concerns relate to the foundational assumption that a formal analysis relies on, e.g. complete and correct variables as well as the atomic execution model. Therefore, sound mathematical characteristics may not be shown on inherently imperfect matters without approximation by the model. Hence, this thesis presents a best-effort model for formal analysis of a context-aware user application acknowledging the context's characteristics.

This model approximates the context and its QoC parameters irreversibly as terms of a predicate at the time of execution by the context dependency operator. This dependency model forces the context-aware user application to evaluate the context before and after a context-aware statement, hence guaranteeing a statement to be executed only in a context. Should the context update in a manner subverting the predicate's outcome during execution, the model was evidently unreasonable, i.e. too far from the reality.

Such modelling of context and QoC parameters approximated as terms of a predicate is supported by alleged separation of concerns in context-aware systems. This separation is between the user application's approximated model on context and the context derivation imperfect view including QoC parameters. Separation also supports context derivation transparency for a context consuming agent facilitating reusability and maintainability as stated in Section 3.1.1. Moreover, the separated views implement different logical topologies in derivation of QoC parameter of trustworthiness.

The context derivation view may require an agent to depend acyclically on several disjoint agents. The logical topology of such a derivation is therefore a polytree. Context derivation in a polytree requires merger and propagation of context and the QoC metadata. This thesis provides the details on how this may be performed on the QoC parameter of trustworthiness, noted a challenging parameter by related work [54] [236]. On the other hand, the user application is concerned with a single context provider providing the situation. Hence, deriving a level of trust possibly by inquiring referrals to ascertain a perception is modelled by a logical topology of a DSPG. This is the original use of subjective logic. With respect to these, this thesis proposes a novel function for merging of context by trustworthiness in a polytree, counting for settings where several providers provide the same context with different values, proposed in Section 5.2.2.

Consequently, Part I of the thesis has presented the following novel contributions:

- Modelling context derivation as a polytree
- Defining a general model for a history of experiences on which calculations of trustworthiness in Subjective logic may be conducted
- Providing a means to calculate a weighted context by trustworthiness

In addition, each of the reprinted publications views some specific problem in a specific setting. Hence, Part I of the thesis also describes how all these aspects relate to each other and provides an overview of the topics discussed; that is a contribution in its own right.

The Success criterions stated in Sections 2.4, 3.4 and 4.4 are addressed forming the basis of the discussion in Chapter 7 that motivates for and against the approaches taken. What publication or in what section these success criterions are addressed is outlined in Table 1 in Chapter 7.

As future perspectives, we envision the pursuit to increased user experience yearning for ever more complex context derivation. This is due to the increasing availability of contexts, personalisation / customisation needs as well as the expanding application domains in form of device mobility, connectivity and context locality. Hence, context is likely to be used in the future in ever more varied and dynamic settings, demanding adaptive means to evaluate it. This brings forward the contributions of this thesis that captures uncertainties on contexts and their derivation as trustworthiness. The extreme of this vision is that in the future, context derivation would constitute the ecosystem populated by adaptive autonomous entities organising themselves to provide contexts desired by user applications [11].

We motivate our views by that abstracting computations to the "cloud" is for real already today. The "social computer" [106] using crowdsourcing is envisioned. Nevertheless, whatever the time frame or reality of realising this extreme vision, the behaviour of a future ubiquitous system is likely to be overly extensive and complex for formalisation as a monolithic structure, hence, requiring means of integration. The reason, as we see it, is that formal methods rely on mathematical correctness and defining correctness is a serious challenge [202]; defining correctness on autonomous agents may just become too difficult. This view also underlines the difficulties of implementing existing formal frameworks on the distributed challenges of autonomous agents populating the transparent distributed system of today and tomorrow. Paradoxically however, at the same time as we envision bold architectures of dynamic uncontrolled behaviour, we are in greater need of formal methods to reason and analyse this complex structure than maybe ever before [52]. As a response to this, this thesis envisions a novel relaxed view on formal methods as to the adaptive parameter of trustworthiness where 'aborting' is not the ultimate fault, but a feature that is inevitable and need to be managed. Consequently, we hope that this thesis provided the reader ideas on how to bring (subjective) "theory into the

unsystematic world of practice" [52] that software is tightly connected with as "software lives in a dirty and imperfect world" [52].

9 Abbreviations and Short Term Definitions

Absolute (experience / opinion): *Boolean valued*
Acquisition model (context): *Models the context's technical derivation*
Actuator: *An agent consuming a formal event and producing an informal event*
Agent: *An entity capable of reasoning. In this thesis typically an application, a user application or an informal being; typically used with context / situation or context-awareness*
Application: *Piece of software that provides for a task optionally consuming context*
Application body: *The logical part of an application*
Approximation: *A generalised view*
Atomic (atomicity): *Without interrupts from the beginning till the end*
Base rate: *A prior probability on uncertainty*
Bayesian: *An evidential probability including uncertainty*
Belief: *The level of warranted expectation that the term meets with expectation*
Bpdf: *Beta Probability Density Function*
Boundary (context) of an application: *All information used within an application but derived from outside is considered context, regardless its origin making the boundary sharp*
Component (software): *An identifiable piece of software that provides a feature and is reusable*
Conceptual model: *A model from the user's point of view that is independent of technological realisation*
Concept: *A modelled construct*
Context consumer: *A context-aware agent whose action depends on the context it acquires*
Context (definition): *Information characterising entities, whose situation is relevant for a context-aware user application, e.g. ring tone level of a phone. See Definition 1*
Context (term): *Used when indifferent whether elementary context or contextual information is meant*
$context_S(t)$: *See system context*
Context acquirer: *The part of an application that acquires context*
Context-aware: *Responsive to some context*
Context-aware system: *The complete architecture incorporating applications and user applications*
Contextual information: *Context derived from elementary context and other contextual information*
Context provider: *An application that provides a context*
Crisp (value / proposition): *A measure without uncertainty*
Correct (correctness): *The mathematical logical property of 'True'*
DAG: *Directed Acyclic Graph*
Demanded situation ($situation_A(t)$): *a set of situations subscribed to by an application at logical time t*
Demonic (choice): *Arbitrary choice, in modelling any choice is possible*

Device: *A piece of equipment*
Disbelief: *The level of warranted expectation that the term fails to meets with expectation*
Dogmatic: *Complete certainty, i.e. no uncertainty*
Dpdf: *Dirichlet Probability Density Function*
DSPG: *Directed Series Parallel Graph*
Element: *A piece of a greater system*
Elementary context: *Raw low-level context captured by a sensor and provided by an application*
Entity: *Something whose situation is sought including the user and application; typically used when indicating a trust relationship*
Environment (context): *The phenomenon that a sensor captures*
Environment (system): *Other agents that may directly or indirectly influence the agent under inspection*
Event (context): *An occurrence of something that gives rise to an experience*
ExaByte: $8*10^{18}$ bits
Experience (in this thesis): *Knowledge of a past event represented by a reading on a scale (sat, unsat)*
Experience-based trust: *A level trustworthiness derived from disjoint experiences*
Explicit context: *Unambiguous configuration inputs to an application*
Formalism: *See formal method*
Formal event: *The occurrence of a well-defined and identified matter*
Formal method: *A methodology of mathematically rigorous techniques enabling tools for the specification, design and verification of software and hardware systems*
Formal mode: *Well-defined mode*
Formal model: *See formal specification*
Formal specification: *A rigorous representation of formal entities and their relations on a model*
fourth: *A function taking the score dimension from the four-tuple of experiences*
Frame of discernment: *The set of possible exclusive outcomes, the valid propositions*
Ignorance: *See uncertainty*
Implicit context: *Ambiguous context describing the environment*
Informal: *Something that destitute a formal (mathematical) representation*
Informal event: *The occurrence of something that is physically identifiable*
Metadata: *Metadata of context is in this thesis the QoC prameters*
Model: *An approximated (generalised) view of the described artefact*
Net situation ($situation_N(t)$): *the relevant context for a user application at logical time t*
Policy-based trust: *A level of (dis)trust determined by defined logical rules (policies).*
Polytree: *A logical topology with no undirected cycles*
Proposition: *A specific outcome within the frame of discernment*

Quality of Context (QoC): *A set of parameters describing the informal quality of a context represented as metadata*

Referral (agent): *A third party whose experiences (opinion) is requested*

Relevant context: *A context used by a user application at time t, the* $situation_N(t)$

Reputation-based trust: *In this thesis, See experience-based trust*

Sensor: *A device capturing an informal phenomenon / event and representing it an formal event, i.e. as a context*

Situation: *A composition of contexts that provides a user application means for context-awareness*

$situation_A(t)$: *See Demanded situation*

$situation_N(t)$: *See Net situation*

Stakeholder: *A thinking being (animals, humans) whose welfare depends on what it is a stakeholder for*

State-space: *A snapshot of variables at a moment*

Stigmergy: *Indirect coordination*

Subjective logic: *A logic on experience-based trust addressing uncertainty*

System context: $context_S(t)$ *set of elementary context or contextual information sensed by the system and valid at logical time t*

Trust (as a term): *General term for the Boolean (un)trusts or level of (un)trustworthiness*

Trusted (trusts) / untrusted (untrusts): *A Boolean level of trustworthiness, i.e. the result of an applied policy defining whether or not to engage in a transaction*

(Un)Trustworthy / (un)trustworthiness: *A level describing the warranted evidence on the trustee behaving according to the expectations of the trustor giving rise for a sense of relative (in)security*

Trustee: *An entity whose trustworthiness is sought, the provider*

Trustor: *An entity with internal goals (intents) whose trust in the trustee is sought, the consumer*

Uncertainty: *The level of 'do not know' with respect to belief and disbelief*

User: *The stakeholder relying on actuation by a user application*

User application: *Piece of software that provides a user means to perform a task optionally consuming a situation*

Virtual object: *A concrete entity whose existence is virtual*

10 References

[1] A. Abdul-Rahman and S. Hailes, "Supporting Trust in Virtual Communities," in *In Proceedings of the 33rd Hawaii International Conference on System Sciences*, 2000.

[2] K. Aberer and Z. Despotovic, "Managing trust in a peer-2-peer information system," in *In Proceedings of the Tenth international Conference on information and Knowledge Management*, 2001, pp. 310-317.

[3] G. Abowd, "Software engineering issues for ubiquitous computing," in *In Proceedings of the 21st international Conference on Software Engineering*, 1999, pp. 75-84.

[4] J-R. Abrial, *Modeling in Event-B: System and Software Engineering*.: Cambridge University Press, 2010.

[5] J.-R. Abrial, *The B-Book: Assigning programs to meanings*. New York, USA: Cambridge University Press, 1996.

[6] J-R. Abrial, S. Schuman, and B. Meyer, "A Specification Language," in *On the Construction of Programs*.: Cambridge University Press, 1980.

[7] G. Agha, "Computing in pervasive cyberspace," *Commun. ACM*, vol. 51, no. 1, pp. 68-70, 2008.

[8] H. Ailisto, P. Alahuhta, V. Haataja, V. Kyllönen, and M. Lindholm, "Structuring Context Aware Applications: Five-Layer Model and Example Case," Position paper Workshop in Ubicomp 2002.

[9] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications Magazine, IEEE*, pp. 102-114, Aug 2002.

[10] H. P. Alesso and C.F Smith, *Thinking on the web*.: Wiley Inc. ISBN-13: 978-0-471-76814-2, 2006.

[11] Amritesh and R. Sarkar, "Conceptualizing 'knowledge organisms' for a sustainable digital knowledge ecosystem," in *3rd IEEE International Conference on Digital Ecosystems and Technologies*, 2009, pp. 247-252.

[12] F. Arbab, "Reo: A Channel-based Coordination Model for Component Composition," *Mathematical Structures in Computer Science*, vol. 14, no. 3, pp. 329-366, 2004.

[13] D. Artz and Y. Gil, "A survey of trust in computer science and the Semantic Web," *Web Semantics: Science, Services and Agents on the World Wide Web, Software Engineering and the Semantic Web*, vol. 5, no. 2, pp. 58-71, 2007.

[14] P. Avesani, P. Massa, and R. Tiella, "A trust-enhanced recommender system application: Moleskiing," in *In Proceedings of the 2005 ACM symposium on Applied computing (SAC '05)*, 2005, pp. 1589-1593.

[15] R. Back, "A Method for Refining Atomicity in Parallel Algorithms. ," in *In Proceedings of the Parallel Architectures and Languages Europe*, 1989, pp. 199-216.

[16] R. Back, "Correctness Preserving Program Refinements: Proof Theory and Applications," Mathematical Centre, Amsterdam, The Netherlands, Mathematical Center Tracts vol: 131, 1980.

[17] R. Back, "On the correctness of refinement steps in program development," Department of Computer Science, University of Helsinki, PhD thesis, Report A-

1978-4 1978.

[18] R. Back, "Refinement calculus, part II: parallel and reactive programs," in *In Proceedings on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness*, 1990, pp. 67-93.

[19] R. Back and R. Kurki-Suonio, "Decentralization of Process Nets with Centralized Control," in *2nd ACM SIGACT-SIGOPS Symp. on Principles of Distributed Computing*, 1983.

[20] R. Back and K. Sere, "Stepwise Refinement of Action Systems," in *In Proceedings of the international Conference on Mathematics of Program Construction, 375th Anniversary of the Groningen University* , 1989, pp. 115-138.

[21] R Back and K. Sere, "Superposition refinement of reactive systems," *Formal Aspects of Computing*, vol. 8, no. 3, pp. 324-346, 1996.

[22] R. Back and J. von Wright, "Refinement Calculus, Part I: Sequential Nondeterministic Programs," in *In Stepwise Refinement of Distributed Systems, Models, Formalisms, Correctness, REX Workshop*, 1990, pp. 42-66.

[23] R. Back and J. von Wright, *Refinement Calculus: A Systematic Introduction*.: Springer, 1998.

[24] R. Back and J. Wright, *Refinement Calculus: a Systematic Introduction*.: Springer-Verlag New York, Inc. , 1998.

[25] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," vol. 2, no. 4, pp. 263 -277, 2007.

[26] G. Banavar et al., "Challenges: an application model for pervasive computing.," in *In Proceedings of the 6th Annual international Conference on Mobile Computing and Networking*, 2000, pp. 266-274.

[27] G. Banavar and A Bernstein, "Software infrastructure and design challenges for ubiquitous computing applications," *Commun. ACM*, vol. 45, no. 12, pp. 92-96, 2002.

[28] B. Barber, *Social Studies of Science*.: Transaction Publishers, 1990.

[29] B. Barber, *The Logic and Limits of Trust*.: Rutgers University Press, 1983.

[30] L. Barkhuus and A. Dey, "Is Context-Aware Computing Taking Control Away from the User? Three Levels of Interactivity Examined," in *In Proceedings of Ubicomp 2003*, 2003, pp. 149 - 156.

[31] L. Bass et al., "A metamodel for the runtime architecture of an interactive system," *ACM SIGCHI Bulletin*, vol. 24, no. 1, pp. 32-37, 1992.

[32] M. Bazire and P. Brézillon, "Understanding Context Before Using It," in *In Modeling and Using Context*, 2005, pp. 29-40.

[33] M. Benerecetti, P. Bouquet, and C. Ghidini, "Contextual Reasoning Distilled," *Journal of Expirimental Artificial Intelligence*, vol. 12, no. 3, pp. 41-67, 2000.

[34] C. Bettini et al., "A survey of context modelling and reasoning techniques," *Pervasive Mob. Comput.*, vol. 6, no. 2, pp. 161-180, 2010.

[35] G. Biegel and V. Cahill, "A Framework for Developing Mobile, Context-aware Applications," in *In Proceedings of the Second IEEE international Conference on Pervasive Computing and Communications (Percom'04)* , 2004.

[36] K. Birman, *Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services*.: Texts in Computer Science, Springer-Verlag London Limited, 2012.

[37] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management ," in *In Proceedings of the 1996 IEEE Symposium on Security and Privacy*, 1996.

[38] C. Bolchini et al., "And what can context do for data?," *Communications of the ACM*, vol. 52, no. 11, pp. 136-140, 2009.

[39] C. Bolchini, C. Curino, E. Quintarelli, F. Schreiber, and L. Tanca, "A data-oriented survey of context models," *SIGMOD Rec.*, vol. 36, no. 4, pp. 19-26, 2007.

[40] P. Bonatti, C. Duma, D. Olemdilla, and N. Shahmehri, "An Integration of Reputation-based and Policy-based Trust Management," in *In Proc. Semantic Web and Policy Workshop*, 2005.

[41] P. Bonatti and P. Samarati, "Regulating service access and information release on the Web," in *In Proceedings of the 7th ACM Conference on Computer and Communications Security*, 2000, pp. 134-143.

[42] G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language User Guide*.: Addison-Wesley, 1998.

[43] P. Bouquet, C. Ghidini, F. Giunchiglia, and E. Blanzieri, "Theories and Uses of Context in Knowledge Representation and Reasoning," *Journal of Pragmatics*, vol. 35, no. 3, pp. 455 -484, 2003.

[44] P. Bouquet, C. Ghidini, F. Giunchiglia, and E. Blanzieri, "Theories and uses of context in knowledge representation and reasoning," *Journal of Pragmatics*, vol. 35, no. 3, pp. 455-484, 2003.

[45] P. Brézillon, "Context dynamic and explanation in contextual graphs," in *In Proceedings of the 4th international and interdisciplinary Conference on Modeling and Using Context* , 2003, pp. 94-106.

[46] P. Brézillon, "Context in problem solving: a survey," *The Knowledge Engineering Review*, vol. 14, pp. 47-80, 1999.

[47] P. Brézillon, "Task-Realization Models in Contextual Graphs," in *Modeling and Using Context*, 2005, pp. 55-68.

[48] T. Broens, "Dynamic context bindings - Infrastructural support for context-aware applications," Univ. of Twente, PhD thesis CTIT Ph.D.-thesis series No. 08-125, ISBN 978-90-75176-47-6 , 2008.

[49] T. Broens, S. Pokraev, M. van Sinderen, J. Koolwaaij, and P. Dockhorn Costa, "Context-aware, ontology-based, service discovery," in *In: European Symposium on Ambient Intelligence*, 2004.

[50] P. Brown, "The stick-e document: a framework for creating context-aware applications," in *Proceedings of the electronic publishing*, 1996, pp. 259 - 272.

[51] P. Brown and G. Jones, "Context-aware Retrieval: Exploring a New Environment for Information Retrieval and Information Filtering.," *Personal Ubiquitous Comput.*, vol. 5, no. 4, pp. 253 - 263, 2001.

[52] M. Broy, "Can Practitioners Neglect Theory and Theoreticians Neglect Practice?," *Computer* , vol. 44, no. 10, pp. 19-24, Oct. 2011.

[53] S. Buchegger and J-Y. Le Boudec, "A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks," in *P2PEcon 2004, Harvard University, Cambridge MA, U.S.A.*, 2004.

[54] T. Buchholz, A. Küpper, and M. Schiffers, "Quality of Context Information: What it is and why we need it," in *In proc. of tge 10th HPOVUA workshop* , Geneva, 2003.

[55] V. Cahill et al., "Using Trust for Secure Collaboration in Uncertain Environments," vol. 2, no. 3, pp. 52-61, 2003.

[56] L. Capra, W. Emmerich, and C. Mascolo, "CARISMA: Context-Aware Reflective mIddleware System for Mobile Applications," *IEEE Trans. Softw. Eng.*, vol. 29, no. 10, pp. 929-945, 2003.

[57] M. Carbone, M. Nielsen, and V. Sassone, "A Formal Model for Trust in Dynamic Networks ," BRICS Report Series Publications. RS-03-4 2003.

[58] C. Castelfranchi and R. Falcone, "Principles of Trust for MAS: Cognitive Anatomy, Social Importance, and Quantification," in *In Proceedings of the 3rd international Conference on Multi Agent Systems*, 1998.

[59] H. Chen, "An Intelligent Broker Architecture for Pervasive Context-Aware Systems ," University of Maryland, Baltimore County, PhD Thesis 2004.

[60] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," 2000.

[61] K. Cheverst, K. Mitchell, and N. Davies, "Investigating Context-aware Information Push vs. Information Pull to Tourists," in *In Proceedings of Mobile HCI 01* , 2001.

[62] B. Christianson and W. Harbison, "Why isn't trust transitive?," in *In Proceedings of the Security Protocols International Workshop*, 1996, pp. 171-176.

[63] H. Clark and S. Brennan, "Grounding in communication," in *Perspectives on socially shared cognition*, L. Resnick, J. Levine, and S. Teasley, Eds.: APA Books, 1991.

[64] D. Clark, C. Partridge, J. Christopher Ramming, and J. Wroclawski, "A knowledge plane for the internet," in *ACM SIGCOMM '03*, 2003, pp. 3-10.

[65] J. Coutaz, J. Crowley, S. Dobson, and D. Garlan, "Context is key," *Commun. ACM*, vol. 48, no. 3, pp. 49-53, 2005.

[66] J. Coutaz and G. Rey, "Foundations for a Theory of Contextors," in *Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces*, 2002, pp. 13-34.

[67] J. Crowley, "Context Driven Observation of Human Activity," in *European Symposium on Ambient Intelligence*, 2003.

[68] J. Crowley, "Situation Models for Observing Human Activity," *ACM Queue Magazine*, vol. 4, no. 6, pp. 35-43, 2006.

[69] J. Crowley, "Social Perception," *Queue 4*, vol. 4, no. 6, pp. 34-43, 2006.

[70] J. Crowley, O. Brdiczka, and P. Reignier, "Learning Situation Models for Understanding Activity ," in *5th International Conference on Development and Learning*, 2006.

[71] J. Crowley, J. Coutaz, G. Rey, and P. Reignier, "Perceptual Components for Context Aware Computing," in *In Proceedings of the 4th international Conference on Ubiquitous Computing*, 2002, pp. 117 - 134.

[72] "Dagstuhl seminar on ubiquitous computing," http://www.inf.ethz.ch/vs/events/dag2001/ visited: 28.7.2010, 2001.

[73] N. Davies and H-W Gellersen, "Beyond Prototypes: Challenges in Deploying Ubiquitous Systems," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 26 - 35, 2002.

[74] A. Dey, "Providing Architectural Support for Context-Aware Applications," Georgia Institute of Technology, PhD Thesis 2000.

[75] A. Dey, "Understanding and Using Context," *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4 - 7, 2001.

[76] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness," in *Workshop on the What, Who, Where, When, Why and How of Context-Awareness*, 2000, In the Workshop on The What, Who, Where, When, and How of Context-Awareness.

[77] A. Dey, G. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," in *Human Computer Interaction.*, 2001, pp. 97 - 166.

[78] A. Dey, J. Mankoff, G. Abowd, and S. Carter, "Distributed mediation of ambiguous context in aware environments," in. *In Proceedings of the 15th Annual ACM Symposium on User interface Software and Technology*, 2002, pp. 121-130.

[79] A. Dey and A. Newberger, "Support for context-aware intelligibility and control," in *In Proceedings of the 27th international Conference on Human Factors in Computing Systems*, 2009, pp. 859-868.

[80] E. W. Dijkstra, *A Discipline of Programming.*: Prentice Hall, 1976.

[81] E. W. Dijkstra, "Guarded commands, nondeterminacy and formal derivation of programs," vol. 18, 8 , no. Commun. ACM, pp. 453-457, 1975.

[82] S. Dobson et al., "A survey of autonomic communications," *ACM Trans. Auton. Adapt. Syst.*, vol. 1, no. 2, pp. 223-259, 2006.

[83] P. Dockhorn Costa, "Architectural support for context-aware applications: from context models to services platforms," Centre for Telematics and Information Technology, University of Twente, PhD thesis http://purl.org/utwente/58357, 2007.

[84] P. Dockhorn Costa, J. Almeida, L. Ferreira Pires, G. Guizzardi, and M van Sinderen, "Towards Conceptual Foundations for Context-Aware Applications," in *In: Proc. of the Third Int'l Workshop on Modeling and Retrieval of Context (MRC'06)*, 2006.

[85] P. Dockhorn Costa, G. Guizzardi, J. Almeida, L. Ferreira Pires, and M. van Sinderen, "Situations in Conceptual Modeling of Context," in *In Proceedings of the 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops (EDOCW '06).*, 2006.

[86] P. Dourish, "What we talk about when we talk about context," *Personal Ubiquitous Comput.*, vol. 8, no. 1, pp. 19-30, 2004.

[87] J. Dunn, *The Concept of Trust in the Politics of John Locke* , JB Schneewind and

Q. Skinner R. Rorty, Ed.: Cambridge University Press, Cambridge, 1984, vol. Philosophy in History.

[88] W. Du and L. Wang, "Context-aware application programming for mobile devices," in *In Proceedings of the 2008 C3S2E Conference*, 2008, pp. 215-227.

[89] E. Emerson and J. Halpern, "Decision procedures and expressiveness in the temporal logic of branching time," *Journal of Computer and System Sciences* , vol. 30, no. 1, pp. 1–24., 1985.

[90] T, Erickson, "Some problems with the notion of context-aware computing," *Commun. ACM 45, 2*, vol. 45, no. 2, pp. 102 - 104, 2002.

[91] J. Eriksson, "Tool-Supported Invariant-Based Programming," Turku Centre for Computer Science, Ph.D. Thesis No 127, 2010.

[92] Rodin: An Open Toolset for Modelling and Reasoning in Event-B, "Abrial, J-R; Butler, M.; Hallerstede, S.; Hoang, T.; Mehta, F.; Voisin, L. ," *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 12, no. 6, pp. 447-466, 2010.

[93] R. Falcone and C. Castelfranchi, *Social trust: a cognitive approach*, Christiano Castelfranchi and Yao-Hua Tan, Ed.: In Trust and deception in virtual societies, Kluwer Academic Publishers, 2001.

[94] A. Ferscha, "Coordination in pervasive computing environments," in *Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 2003, pp. 3- 9, 9-11.

[95] A. Ferscha, C. Holzmann, and S. Oppl, "Context awareness for group interaction support," in *In Proceedings of the Second international Workshop on Mobility Management &Amp; Wireless Access Protocols*, 2004.

[96] J. Filho, A. Dia Miron, I. Satoh, J. Gensel, and H. Martin, "Modeling and Measuring Quality of Context Information in Pervasive Environments," in *IEEE International Conference on Advanced Information Networking and Applications*, 2010, pp. 690-697.

[97] A. Fitzpatrick, G. Biegel, S. Clarke, and V. Cahill, "Towards a Sentient Object Model," in *Workshop on Engineering Context-Aware Object Oriented Systems and Environments (OOPSLA/ECOOSE'02)*, 2002.

[98] P. Flocchini and F. Luccio, "Routing in Series Parallel Networks," *Theory of Computing Systems*, vol. 2, no. 36, pp. 137-157, 2003.

[99] B. Friedman, P. Khan, and D. Howe, "Trust online," *Commun. ACM*, vol. 43, no. 12, pp. 34-40, 2000.

[100] A. Fuchs, S. Gürgens, and C. Rudolph, "A Formal Notion of Trust - Enabling Reasoning about Security Properties," in *IFIPTM 2010*, 2010, pp. 200-215.

[101] A. Fuchs, S. Gürgens, and C. Rudolph, "Formal Notions of Trust and Confidentiality - Enabling Reasoning about System Security," *Journal of Information Processing*, vol. 19, pp. 274-291, 2011.

[102] D. Gambetta, "Can We Trust Trust?," in *Trust: Making and Breaking Cooperative Relations*.: Department of Sociology, University of Oxford, chapter 13, pp. 213-237, 2000.

[103] H. Gellersen, A. Schmidt, and M. Beigl, "Multi-sensor context-awareness in

mobile devices and smart artifacts," *Mob. Netw. Appl.* , vol. 7, no. 5, pp. 341-351, 2002.

[104] C. Ghidini and F. Giunchiglia, "Local Models Semantics, or contextual reasoning = locality + compatibility," *Artificial Intelligence*, vol. 127, no. 2, pp. 221-259, 2001.

[105] F. Giunchiglia, "Contextual reasoning," *Epistemologia (Special Issue on I Linguaggi e le Macchine)*, vol. 16, pp. 345–364, 1993.

[106] F. Giunchiglia and D. Robertson, "The Social Computer: Combining Machine and Human Computation," Ingegneria e Scienza dell'Informazione, University of Trento., Technical Report DISI-10-036 2010.

[107] J. Golbeck, *Computing with social trust*, J. Golbeck, Ed.: Springer, 2009.

[108] A. Gonzalez and R. Ahlers, "Context-based representation of intelligent behavior in training simulations," *Trans. Soc. Comput. Simul. Int.*, vol. 15, no. 4, pp. 153-166, 1998.

[109] T. Grandison, "Trust Management for Internet Applications ," Imperial College London, PhD Thesis 2003.

[110] T. Grandison and M. Sloman, "A Survey of Trust in Internet Applications," *IEEE Communications Surveys and Tutorials*, vol. 3, no. 4, 2000.

[111] P. Gray and D Salber, "Modelling and Using Sensed Context Information in the Design of Interactive Applications," in *In Proceedings of the 8th IFIP international Conference on Engineering For Human-Computer interaction*, vol. LNCS vol. 2254, 2001, pp. 317-336.

[112] S. Greenberg, "Context as a dynamic construct," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 257-268, 2001.

[113] M. Grossmann, N. Hönle, C. Lübbe, and H. Weinschrott, "An abstract processing model for the quality of context data," in *In Proceedings of the 1st international conference on Quality of context (QuaCon'09)*, 2009, pp. 132-143.

[114] Trusted Computing Group. (2011) TPM Main Specification Level 2 Version 1.2, Revision 116. [Online]. http://www.trustedcomputinggroup.org/resources/tpm_main_specification

[115] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *Proceedings of the 13th International Conference on World Wide Web* , 2004, pp. 403–412.

[116] G. Guizzardi, "Ontological foundations for structural conceptual models," University of Twente, PhD Thesis 2005.

[117] T. Gu, H. Pung, and D. Zhang, "A service-oriented middleware for building context-aware services.," *Journal of Network and Computer Applications* , vol. 1, no. 28, pp. 1 -18, 2005.

[118] R. Gustavsen, "Condor - an application framework for mobilty-based context-aware applications," in *Workshop on concepts and models for ubiquitous computing*, 2002.

[119] J. Gwizdka, "What' s in the Context? ," in *Workshop The What, Who, Where, Why and How of Context-Awareness*, 2000.

[120] K. Henricksen, "A framework for context-aware pervasive computing applications," School of Information Technology and Electrical Engineering, The University of Queensland , PhD Thesis 2003.

[121] K. Henricksen and J. Indulska, "A Software Engineering Framework for Context-Aware Pervasive Computing," in *In Proceedings of the Second IEEE international Conference on Pervasive Computing and Communications* , 2004.

[122] K. Henricksen and J. Indulska, "Developing context-aware pervasive computing applications: Models and approach.," *Pervasive Mob. Comput.*, vol. 2, no. 1, pp. 37-64, 2006.

[123] K. Henricksen and J. Indulska, "Modelling and Using Imperfect Context Information," in *In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW '04)*, 2004, p. 200.

[124] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam, "Middleware for Distributed Context-Aware Systems," in *In OTM Conferences*, 2005, pp. 846-863.

[125] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Infrastructure for Pervasive computing: Challenges," in *Workshop on Pervasive Computing INFORMATIK 01*, Vienna, 2001.

[126] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling Context Information in Pervasive Computing Systems," in *In Proceedings of the First international Conference on Pervasive Computing* , 2002, pp. 167-180.

[127] K. Henricksen, S. Livingstone, and J. Indulska, "Towards a hybrid approach to context modeling, reasoning and interoperation.," in *Proceedings of the First International Workshop on Advanced Context Modelling, Reasoning and Management*, 2004, pp. 54-61.

[128] C. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576 - 580, 583, October 1969.

[129] C. Hoare, "Communicating sequential processes," *Communications of the ACM*, vol. 21, no. 8, pp. 666–677, 1978.

[130] T. Hofer et al., "Context-Awareness on Mobile Devices - the Hydrogen Approach," in *International Conference on System Sciences (HICSS'03*, 2003.

[131] M. Hollis, *Trust within reason*.: Cambridge University Press, 1998.

[132] J. Hong and J. Landay, "An infrastructure approach to context-aware computing," *Hum.-Comput. Interact. 16, 2 (Dec. 2001)*, vol. 16, no. 2, pp. 287-303, 2001.

[133] J. Hong, E. Suh, and S. Kim, "Context-aware systems: A literature review and classification," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 8509-8522, 2009.

[134] J. Hong, E-H. Suh, J. Kim, and S. Kim, "Context-aware system for proactive personalized service based on context history," *Expert Syst. Appl.* , vol. 36, no. 4, pp. 7448-7457, 2009.

[135] M. Atrey, P. Hossain and A. El Saddik, "Modeling Quality of Information in Multi-sensor Surveillance Systems," in *IEEE 23rd international Conference on Data Engineering Workshop*, 2007.

[136] R. Hull, P. Neaves, and J. Bedford-Roberts, "Towards Situated Computing," in *In*

Proceedings of the 1st IEEE international Symposium on Wearable Computers , 1997, p. 146..

[137] S. Hänninen, J. Järvenpää, M. Reunanen, and J. Suominen, "Mekatronisten komponenttien ja laitteiden vikaantuminen," Suomen metalliteollisuuden keskusliitto MET , Tekninen tiedoitus 15 Isbn 951-817-479-2, 1990.

[138] IDC, "As the Economy Contracts, the Digital Universe Expands," IDC, White Paper 2009.

[139] J. Indulska and P. Sutton, "Location management in pervasive systems," in *In Proceedings of the Australasian information Security Workshop Conference on ACSW Frontiers* , vol. 21, 2003, pp. 141 - 151.

[140] R. Jansen, T. Kaminski, F. Korsakov, A. Saint Croix, and D. Selifonov, "A Priori Trust Vulnerabilities in EigenTrust," Technical report 2008.

[141] A. Jøsang, "A logic for uncertain probabilities," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 9, no. 3, pp. 279-311., 2001.

[142] A. Jøsang, "Artificial Reasoning with Subjective Logic," in *Second Australian Workshop on Commonsense Reasoning*, 1997.

[143] A. Jøsang, "Conditional Reasoning with Subjective Logic," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 1, no. 15, pp. 5-38, 2008.

[144] A. Jøsang, "Subjective Logic," Draft book Available at: http://persons.unik.no/josang/papers/subjective_logic.pdf , visited 26.10.2010, Unpublished.

[145] A. Jøsang, "Trust-Based Decision Making for Electronic Transactions," in *Proceedings of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC'99)*, 1999.

[146] A. Jøsang, T. Ažderska, and S. Marsh, "Trust Transitivity and Conditional Belief Reasoning," in *In proceedings of IFIPTM 2012*, 2012, pp. 68-83.

[147] A. Jøsang and J. Haller, "Dirichlet Reputation Systems ," in *The Second International Conference on Availability, Reliability and Security* , 2007, pp. 112-119.

[148] A. Jøsang, R. Hayward, and S. Pope, "Trust network analysis with subjective logic," in *In Proceedings of the 29th Australasian Computer Science Conference*, vol. 48, 2006, pp. 85-94.

[149] A. Jøsang and R. Ismail, "The beta reputation system," in *In Proceedings from the 15th Bled Conference on Electronic Commerce*, 2002.

[150] A. Jøsang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decis. Support Syst.*, vol. 43, no. 2, pp. 618-644, 2007.

[151] A. Jøsang and D. McAnally, "Multiplication and Comultiplication of Beliefs," *International Journal of Approximate Reasoning* , vol. 38, no. 1, pp. pp.19-51, 2004.

[152] A. Jøsang and S. Pope, "Semantic constraints for trust transitivity," in *In Proceedings of the 2nd Asia-Pacific conference on Conceptual modelling* , 2005.

[153] A. Jøsang, S. Pope, and M. Daniel, "Conditional Deduction Under Uncertainty," in *Proceedings of the 8th European Conference on Symbolic and Quantitative*

Approaches to Reasoning with Uncertainty, 2005, pp. 824-835.

[154] A. Jøsang, S. Pope, and S. Marsh, "Exploring Different Types of Trust Propagation," in *Proceedings of the 4th International Conference on Trust Management (iTrust'06)*, 2006, pp. 179-192.

[155] S. Kamvar, M. Schlosser, and H. Garcia-Molina, "The Eigentrust algorithm for reputation management in P2P networks," in *In Proceedings of the 12th international Conference on World Wide Web*, 2003, pp. 640-651.

[156] G. Kappel, B. Proll, W. Retschitzegger, and W. and Schwinger, "Customisation for ubiquitous web applications: a comparison of approaches," *Int. J. Web Eng. Technol.*, vol. 1, no. 1, pp. 79-111, 2003.

[157] J. Kephart and D. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41-50, 2003.

[158] Y. Kim and K. Lee, "A Quality Measurement Method of Context Information in Ubiquitous Environments ," in *International Conference on Hybrid Information Technology*, 2006, pp. 576-581.

[159] J. Kjeldskov and C. Graham, "A Review of Mobile HCI Research Methods," , 2003.

[160] T. Korsgaard and C. Damsgaard Jensen, "Reengineering the Wikipedia for Reputation," *In Electronic Notes in Theoretical Computer Science*, vol. 244, pp. 81-94, 2009.

[161] M. Krause and I. Hochstatter, "Challenges in Modelling and Using Quality of Context (QoC)," in *Mobility Aware Technologies and Applications*, 2005, pp. 324 - 333.

[162] K. Krukow, "Towards a theory of trust for the global ubiquitous computer," University of Aarhus, PhD Thesis 2006.

[163] K. Krukow, M. Nielsen, and V. Sassone, "Trust models in ubiquitous computing," *Philos Transact A Math Phys Eng Sci.*, vol. 366, pp. 3781-3793, 2008.

[164] S. Lahlou, M. Langheinrich, and C. Röcker, "Privacy and trust issues with invisible computers," *Commun. ACM*, vol. 48, no. 3, pp. 59 - 60, 2005.

[165] R. Lange et al., "Making the World Wide Space happen: New challenges for the Nexus context platform," in *In Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications (PERCOM '09)*, 2009, pp. 1-4.

[166] R. Lange et al., "Making the World Wide Space happen: New challenges for the Nexus context platform," in *In Proceedings of the 2009 IEEE International Conference on Pervasive Computing and Communications (PERCOM '09).*, 2009.

[167] J. Laprie, "Dependable Computing and Fault Tolerance: Concepts and terminology ," in *In Proc. 15th IEEE Int. Symp. on Fault-Tolerant Computing*, 1985.

[168] W. Lee, "Deploying personalized mobile services in an agent-based environment," vol. 32, no. 4, pp. 1194-1207, 2007.

[169] H. Lei, D. Sow, J. Davis, G. Banavar, and M. Ebling, "The design and applications of a context service," *SIGMOBILE Mob. Comput. Commun.*, vol. 6, no. 4, pp. 44-55, 2002.

[170] H. Lieberman and T. Selker, "Out of context: computer systems that adapt to, and learn from, context," vol. IBM Syst. J. 39, no. 3 - 4, pp. 617-632, 2000.

[171] H. Li, J. Salomaa, M. Jian, and Y. Kuifei, "Research on Context-Aware Mobile Computing," in *In Proceedings of the 22nd international Conference on Advanced information Networking and Applications*, 2008, pp. 24-30.

[172] S. Loke, "Building Taskable Spaces over Ubiquitous Services," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. pp. 72-78, 2009.

[173] S. Loke, "Incremental awareness and compositionality: A design philosophy for context-aware pervasive systems," *Pervasive Mob. Comput.*, vol. 6, no. 2, pp. 239-253, 2010.

[174] S. Loke, "Logic programming for context-aware pervasive computing: language support, characterizing situations, and integration with the web ," in *Proceedings of 2004 IEEE/WIC/ACM International Conference on Web Intelligence*, 2004, pp. 44-50.

[175] K. Mani Chandy, *Parallel Program Design: a Foundation*.: Addison-Wesley Longman Publishing Co., Inc., 1988.

[176] A. Manzoor, H. Truong, and S. Dustdar, "On the Evaluation of Quality of Context," in *In Proceedings of the 3rd European Conference on Smart Sensing and Context*, 2008, pp. 140-153.

[177] S. Marsh, "Formalizing Trust as a Computational Concept," University of Stirling, Department of Computer Science and Mathematics, PhD thesis 1994.

[178] S. Marsh and P. Briggs, *Computing with social trust*, J. Golbeck, Ed.: Springer, 2009, Chapter 2.

[179] S. Marsh and P. Briggs, "Short paper: Defining and Investigating Device Comfort," in *4th IFIP WG 11.11 International Conference on Trust Management*, 2010, pp. 17-24.

[180] S. Marsh, P. Briggs, K. El-Khatib, Esfandiari B., and J. Stewart, "Defining and Investigating Device Comfort ," *Journal of Information Processing*, vol. 19, pp. 231-252, July 2011.

[181] S. Marzano and E. Aarts, *The New Everyday View on Ambient Intelligence*.: Uitgeverij 010 Publishers, 2003, ISBN: 9064505020.

[182] P. Massa and P. Avesani, "Trust Metrics on Controversial Users: Balancing Between Tyranny of the Majority and Echo Chambers," *International Journal on Semantic Web and INformation Systems*, vol. 3, no. 1, 2007.

[183] R. Masuoka, B. Parsia, and Y. Labrou, "Task Computing — The Semantic Web Meets Pervasive Computing ," in *Proc. 2nd Int'l Semantic Web Conf. ,* 2003, pp. pp. 866–881.

[184] J. McCarthy, "Notes on formalizing context," in *In Proceedings of the 13th international Joint Conference on Artifical intelligence*, 1993, pp. 550-560.

[185] J. McCarthy and P. Hayes, "Some Philosophical Problems from the Stand-point of Artificial Intelligence," in *Machine Intelligence 4*.: Edinburgh University Press, 1969, pp. 463-502.

[186] A McIver and C. Morgan, *Abstraction, Refinement and Proof for Probabilistic Systems*.: Springer Monographs in Computer Science, 2005.

[187] H. McKnight and N. Chervaney, "The Meanings of Trust," Technical Report Working Paper Series 96-04 1996.

[188] (2011, July) Merriam Webster Online Dictionary. [Online]. http://www.merriam-webster.com/dictionary/context

[189] R. Milner, *A Calculus of Communicating Systems*.: Springer Verlag, 1980.

[190] R. Milner, J. Parrow, and D. Walker, "A calculus of mobile processes, I," *Inf. Comput.* , vol. 100, no. 1, pp. 1-40, 1992.

[191] T. Moran and P. Dourish, "Introduction to this special issue on context-aware computing," *Hum.-Comput. Interact.*, vol. 16, no. 2, pp. 87-95, 2001.

[192] C. Morgan, *Programming from Specifications*.: Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[193] C. Morgan and A. McIver, "pGCL: Formal reasoning for random algorithms," *South African Computer Journal*, vol. 22, pp. 14-27, 1999.

[194] C. Morgan, A. McIver, and Seidel. K., "Probabilistic Predicate Transformers," *ACM Transactions on Programming Languages and Systems*, 1995.

[195] G. Mostefaoui, J. Pasquier-Rocha, and P. Brezillon, "Context-Aware Computing: A Guide for the Pervasive Computing Community," in *IEEE/ACS International Conference on Pervasive Services*, 2004, pp. 39-48.

[196] L. Mui, M. Mohtashemi, C. Ang, P. Szolovits, and A. Halberstadt, "Bayesian Ratings in Distributed Systems: Theories, Models, and Simulations," MIT LCS Memorandum 2001.

[197] L. Mui, M. Mohtashemi, and A Halberstadt, "A Computational Model of Trust and Reputation for E-businesses," in *In Proceedings of the 35th Annual Hawaii international Conference on System Sciences Hicss*, 2002.

[198] P. Naur, "Formalization in program development," *BIT Numerical Mathematics*, vol. 22, no. 4, pp. 437 - 453, Dec. 1982.

[199] A. Oulasvirta, "Grounding the innovation of future technologies," *An Interdisciplinary Journal on Humans in ICT Environments*, vol. 1, no. 1, pp. 58 - 75, 2005.

[200] A. Padovitz, S. Loke, and A. Zaslavsky, "Multiple-Agent Perspectives in Reasoning About Situations for Context-Aware Pervasive Computing Systems," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 38, no. 4, pp. 729-742, 2008.

[201] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Technical Report. Stanford InfoLab. 1999.

[202] D. Parnas, "Really Rethinking 'Formal Methods," *Computer*, vol. 43, no. 1, pp. 28-34, 2010.

[203] J. Pascoe, "Adding Generic Contextual Capabilities to Wearable Computers," in *In Proceedings of the Second International Symposium on Wearable Computers*, 1998.

[204] J. Pearl, "Reasoning with Belief Functions: An Analysis of Compatibility," *The International Journal of Approximate Reasoning*, vol. 4, no. 5/6, pp. 363-389, 1990.

[205] M. Perttunen, J. Riekki, and O. Lassila, "Context Representation and Reasoning in Pervasive Computing: a Review," *International Journal of Multimedia and Ubiquitous Engineering*, vol. 4, no. 4, 2009.

[206] L. Petre, "Modeling with Action Systems ," Turku Centre for Computer Science, Turku, PhD Thesis, TUCS Dissertations 69 2005.

[207] A. Pnueli, "The temporal logic of programs," in *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, 1977, pp. 46-57.

[208] S. Pope and A. Jøsang, "Analysis of Competing Hypothesis using Subjective Logic ," in *Proceedings of the 10th International Command and Control Research Technology Symposium* , 2005.

[209] P. Prekop and M. Burnett, "Activities, context and ubiquitous computing," *Computer Communications*, vol. 26, no. 11, pp. 1168-1176, 2003.

[210] D. Quercia, S. Hailes, and L. Capra, "B-Trust: Bayesian trust framework for pervasive computing," in *In: Trust Management: Proceedings of the 4th International Conference, iTrust 2006.* , 2006, pp. 298-312.

[211] A. Ranganathan, J. Al-Muhtadi, and R. Campbell, "Reasoning about Uncertain Contexts in Pervasive Computing Environments," *IEEE Pervasive Computing*, vol. 3, no. 2, pp. 66-70, 2004.

[212] R. Reiter, "The Situation Calculus Ontology," *Electronic News Journal on Reasoning about Actions and Change*, vol. 2, p. NIL , 1997.

[213] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman, "Reputation systems," *Commun. ACM* , vol. 43, no. 12, pp. 45-48, 2000.

[214] G. Rey and J. Coutaz, "The Contextor Infrastructure for Context-Aware Computing," in *Component-oriented Approaches to Context-aware Computing ECOOP'04*, 2004.

[215] G. Rey and J. Coutaz, "The contextor infrastructure for context-aware computing," in *Workshop on Component-oriented Approaches to Context-aware Computing*, 2004.

[216] G. Roman, C. Julien, and J. Payton, "A formal treatment of context-awareness.," in *Fundamental Approaches to Software Engineering FASE'04*, 2004.

[217] S. Ruohomaa, L. Kutvonen, and E. Koutrouli, "Reputation Management Survey ," in *The Second International Conference on Availability, Reliability and Security* , 2007, pp. 103-111.

[218] N. Ryan, J. Pascoe, and D. Morse, "Enhanced Reality Fieldwork: the Context-aware Archaeological Assistant," British Archaeological Reports, Oxford, 1998 1998.

[219] McKeever S., J. Ye, Coyle L., and S. Dobson, "A Multilayered Uncertainty Model for Context Aware Systems," in *Pervasive 2008*, 2008.

[220] J. Sabater-Mir and M. Paolucci, "On representation and aggregation of social evaluations in computational trust and reputation models," vol. 46, no. 3, pp. 458-483, 2007.

[221] J. Sabater and C. Sierra, "Social ReGreT, a reputation model based on social relations," *SIGecom Exch.*, vol. 3, no. 1, pp. 44-56, Dec. 2001.

[222] M. Satyanarayanan, "Pervasive Computing: Vision and Challenges," *IEEE Personal Communications*, vol. 8, pp. 10 - 17, 2001.

[223] W. Schilit, "A System Architecture for Context-Aware Mobile Computing," Columbia University, PhD Thesis 1995.

[224] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," *IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, pp. 89-101, 1994.

[225] A. Schmidt, "A Layered Model for User Context Management with Controlled Aging and Imperfection Handling ," in *Proceedings of the 2nd InternationalWorkshop on Modeling and Retrievalof Context*, 2006, pp. 86-100.

[226] A. Schmidt et al., "Advanced Interaction in Context," in *In Proceedings of the 1st international Symposium on Handheld and Ubiquitous Computing*, 1999.

[227] R. Schmohl and U. Baumgarten, "Context-aware Computing: a Survey Preparing a Generalized Approach ," in *Proc. of the Int. MultiConference of Engineers and Computer Scientists*, 2008.

[228] J. Schneider, G. Kortuem, J. Jager, S. Fickas, and Z. Segall, "Disseminating Trust Information in Wearable Communities," *Personal Ubiquitous Computing*, vol. 4, no. 4, pp. 245-248, Jan. 2000.

[229] B. Schroeder, E. Pinheiro, and W-D. Weber, "DRAM errors in the wild: a large-scale field study," in *In Proceedings of the eleventh international joint conference on Measurement and modeling of computer systems (SIGMETRICS '09)*, 2009, pp. 193-204.

[230] E. Sekerinski and K. Sere, "A Theory of Prioritizing Composition," *The Computer Journal*, vol. 39, no. 8, pp. 701-712, 1996.

[231] K. Sentz and S. Ferson, "Combination of Evidence in Dempster–Shafer Theory," SAND 2002-0835 2002.

[232] K. Sere, "Stepwise derivation of parallel algorithms," Åbo Akademi, Department of computer science, PhD Thesis ISBN: 951-649-748-9, 1990.

[233] K. Sere and M. Waldén, "Data Refinement of Remote Procedures," *Formal Aspects of Computing*, vol. 12, no. 4, pp. 278-297, 2000.

[234] M. Shaw, "Beyond Objects: A Software Design Paradigm Based on Process Control," *ACM SIGSOFT Software Engineering Notes* , vol. 20, pp. 27-38, 1995.

[235] M. Shaw and D. Garlan, *Software Architecture*, Mona Pompili, Ed. New Jersey: Prentice-Hall Inc., 1996.

[236] K. Wegdam, M. Sheikh and M. van Sinderen, "Middleware Support for Quality of Context in Pervasive Context-Aware Systems ," in *In: Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, 2007.

[237] K. Sheikh, M. Wegdam, and M. Sinderen, "Quality-of-Context and its use for Protecting Privacy in Context Aware Systems," *Journal of Software*, vol. 3, no. 3, pp. 83 - 93, 2008, Available at: http://ojs.academypublisher.com/index.php/jsw/article/view/1939. Date acc.

[238] A. Soylu, P. De Causmaecker, and P. Desmet, "Context and Adaptivity in Pervasive Computing Environments: Links with Software Engineering and

Ontological Engineering," *Journal of Software*, vol. 4, no. 9, pp. 992-1013, 2009.

[239] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey," in *Workshop on Advanced Context Modelling, Reasoning and Management*, 2004.

[240] R. Studer, V. Benjamins, and D. Fensel, "Knowledge engineering: Principles and methods," *Data and Knowledge Engineering*, vol. 25, no. 1-2, pp. 161-197, 1998.

[241] C. Szyperski, *Component Software*.: Addison-Wesley, 1998.

[242] C. Szyperski, D. Gruntz, and S. Murer, *Component Software – Beyond Object-Oriented Programming – Second Edition*.: Addison-Wesley and ACM Press, 2002.

[243] R. Taylor et al., "A component- and message-based architectural style for GUI software ," *IEEE Transactions on Software Engineering*, vol. 22, no. 6, pp. 390-406, 1996.

[244] L. Teacy, J. Patel, N. Jennings, and M. Luck, "Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model," in *In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS '05)*, 2005, pp. 997-1004.

[245] F. Turkmen, B. Crispo, and P. Mazzoleni, "A service-based context management framework for cross-enterprise collaboration," in *In Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010.

[246] X. Wang, D. Zhang, T. Gu, and H. Pung, "Ontology Based Context Modeling and Reasoning using OWL," in *In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004.

[247] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The Active Badge Location System," *ACM Transactions on Informations Systems*, vol. 10, 1992.

[248] A. Ward, A. Jones, and A Hopper, "A new location technique for the active office," *IEEE Personal Communications* , vol. 4, no. 5, pp. 42 - 47, 1997.

[249] S. Weeks, "Understanding Trust Management Systems ," in *In Proceedings of the 2001 IEEE Symposium on Security and Privacy*, 2001.

[250] M. Wegdam, "Dynamic Reconfiguration and Load Distribution in Component Middleware," University of Twente, Enschede, PhD thesis 2003.

[251] M. Weiser, "Hot Topics: Ubiquitous Computing," *IEEE Computer*, October 1993.

[252] M. Weiser, "Some Computer Science Problems in Ubiquitous Computing," *Communications of the ACM*, July 1993.

[253] M. Weiser, "The Computer of the Twenty-First Century," *Scientific American*, September 1991.

[254] M. Weiser and J. Brown, "Designing Calm Technology," Xerox PARC, 1995.

[255] A. West, S. Kannan, I. Lee, and O. Sokolsky, "An evaluation framework for reputation management systems," in *Trust Modeling and Management in Digital Environments: From Social Concept to System Development*.: IGI Global, 2010.

[256] A. Whitby, A. Josang, and J. Indulska, "Filtering out unfair ratings in bayesian reputation systems," in *Proceedings of the Third International Joint Conference on Autonomous Agenst and Multi Agent S*, 2004, pp. 106-117.

[257] C. Villalonga, D. Roggen, C. Lombriser, P. Zappi, and G. Tröster, "Bringing

quality of context into wearable human activity recognition systems.," in *In Proceedings of the 1st international Conference on Quality of Context*, 2009, pp. 164-173.

[258] R. Wille, *Restructuring lattice theory: an approach based on hierarchies of concepts*.: Dordrecht-Boston , 1982.

[259] T. Winograd, "Architectures for context," in *Hum.-Comput. Interact.*, 2001, pp. 401-419.

[260] M. Winslett et al., "Negotiating Trust on the Web," *IEEE Internet Computing* , vol. 6, no. 6, pp. 30-37, 2002.

[261] W. Woods, "Taxonomic lattice structures for situation recognition," in *In Proceedings of the 1978 workshop on Theoretical issues in natural language processing (TINLAP '78)*, 1978, pp. 33-41.

[262] L. Xiong and L. Liu, "PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities," *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 7, pp. 843-857, 2004.

[263] L. Yan, "Systematic design of ubiquitous systems," Åbo Akademi / Turku Center for Computer Science, PhD Thesis no. 70 2005.

[264] K. Yang and A. Galis, "Policy-Driven Mobile Agents for Context-Aware Service in Next Generation Networks," in *In Proceedings of the 5th International workshop on Mobile Agents for Telecommunication Applications MATA03*, 2003.

[265] Z Yan and S. Holtmanns, "Trust Modeling and Management: from Social Trust to Digital Trust," in *Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions*, R. Subramanian, Ed.: IGI Global, 2007.

[266] Z. Yan and C. Prehofer, "Autonomic Trust Management for a Component-Based Software System," *IEEE Transactions on Dependable and Secure Computing*, vol. 8, no. 6, pp. 810-823, 2011.

[267] L. Yan and K. Sere, "A Formalism for Context-Aware Mobile Computing," in *Third International Symposium on Parallel and Distributed Computing*, 2004, pp. 14-21.

[268] J. Ye, L. Coyle, S. Dobson, and P. Nixon, "Representing and manipulating situation hierarchies using situation lattices," *Revue d'Intelligence Artificielle*, vol. 22, no. 5, pp. 647-667, 2008.

[269] J. Ye, L. Coyle, S. Dobson, and P. Nixon, "Using situation lattices to model and reason about context," in *In Proceedings of MRC 2007* , 2007, pp. 1-12.

[270] J. Ye, S. Dobson, and S. McKeever, "Situation identification techniques in pervasive computing: a review," *Pervasive and Mobile Computing*, 2011.

[271] X. Ying and X. Fu-yuan, "Research on Context Modeling Based on Ontology," in *Computational Intelligence for Modelling, Control and Automation, 2006 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, 2006.

[272] B. Yu and M. Singh, "An evidential model of distributed reputation management," in *In Proceedings of the first international joint conference on Autonomous agents and multiagent systems:* , DOI=10.1145/544741.544809, 2002, pp. 294-301.

[273] L. Zadeh, "Reviews of Books: A Mathematical Theory of Evidence," *The AI*

Magazine v5, pp. 81-83, 1984.

[274] E. Zelkha and B. Epstein, "From Devices to "Ambient Intelligence," Digital Living Room Conference, June 1998.

[275] H. Zemanek, "Abstract Architecture, General concepts for systems design," Paper for the Winterschool on Abstract Software Specification at the Danish University of Technology, Lecture Notes in Computer Science 86/1980 ISBN 978-3-540-10007-2, 1980.

[276] A. Zimmermann, A. Lorenz, and R. Oppermann, "An operational definition of context," in *In Proceedings of the 6th international and interdisciplinary Conference on Modeling and Using Context* , 2007, pp. 558-571.

[277] P. Öztürk and A. Aamodt, "Towards a model of context for case-based diagnostic problem solving," in *Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context*, 1997, pp. 198-208.

Complete List of Original Publications

1. Mats Neovius, "*An Abstract Model for Incentive-Enhanced Trust in P2P Networks*". In: Tomoya Enokido, Lu Yan, Bin Xiao, Daeyoung Kim, Yuanshun Dai, Laurence T. Yang (Eds.), *Embedded and Ubiquitous Computing - EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES, Nagasaki, Japan, December 6-9, 2005.* , Lecture Notes in Computer Science vol. 3823, 602 - 611, Springer Berlin / Heidelberg, 2005.
2. Kaisa Sere, Lu Yan, Mats Neovius, "*Dependability Challenge in Ubiquitous Computing*". In: *Proceedings of the International Workshop on Software Engineering Challenges for Ubiquitous Computing (SEUC 2006), June 1-2, 2006 - Lancaster, UK*, 2006.
3. Mats Neovius, Lu Yan, "*A Design Framework for Wireless Sensor Networks*". In: Khaldoun Al Agha (Ed.), *Ad-Hoc Networking: IFIP 19th World Computer Congress, TC-6, IFIP Interactive Conference on Ad-Hoc Networking, August 20-25, 2006, Santiago, Chile* , IFIP International Federation for Information Processing 212, 119 - 127, Springer, 2006.
4. Mats Neovius, Kaisa Sere, Lu Yan, Manoranjan Satpathy, "*A Formal Model of Context-Awareness and Context-Dependency*". In: Van Hung Dang, Pandya Paritosh (Eds.), *Proceedings of the fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06), 2006.*, 177 - 185, IEEE Computer Society Press, 2006.
5. Pontus Boström, Mats Neovius, Ian Oliver, Marina Waldén, "*Formal Transformation of Platform Independent Models into Platform Specific Models*". In: Jacques Julliand, Olga Kouchnarenko (Eds.), *B 2007: Formal Specification and Development in B, 7th International Conference of B Users, Besancon, France, January 7-19, 2007, Proceedings*, Lecture Notes in Computer Science vol. 4355, 186-200, Springer-Verlag, 2007.
6. Fredrik Degerlund, Mats Neovius, Kaisa Sere, "*A Framework for Formal Reasoning about Distributed Webs of Trust*". In: Olaf Owe Einar Broch Johnsen, Gerardo Schneider (Eds.), *Proceedings of the 19th Nordic Workshop on Programming Theory*, Universitetet i Oslo - Institutt for informatikk Research Report 366, 78-80, University of Oslo, 2007.
7. Mats Neovius, Fredrik Degerlund, "*Extending Dependability to Include User-Specific Trust*". In: Kaisa Sere Luigia Petre, Einar Broch Johnsen (Eds.), *NODES 07 - NOrdic workshop and doctoral symposium on*

DEpendability and Security, Oslo, Norway, October 2007, Abstracts, Åbo Akademi Reports on Computer Science & Mathematics Ser. B. No. 37, Åbo Akademi University, 2007.

8. Mats Neovius, Fredrik Degerlund, Lu Yan, "*Forming a Context-Sensitive Web of Trust by Relying on Sentimentally Like-Minded*". *International Journal of Pervasive Computing and Communications* 4(1), 92-109, 2008.
9. Mats Neovius, Kaisa Sere, "*Formal Modular Modelling of Context-Awareness*". In: Frank S. de Boer, Marcello M. Bonsangue, Eric Madelain (Eds.), *Formal Methods for Components and Objects, 7th International Symposium, FMCO 2008, Revised Lectures*, 102-118, Lecture Notes in Computer Science vol. 5751, 2008.
10. Petter Sandvik, Mats Neovius, "*The Distance-Availability Weighted Piece Selection Method for BitTorrent: A BitTorrent Piece Selection Method for On-Demand Streaming*". In: Antonio Liotta, Nick Antonopoulos, George Exarchakos, Takahiro Hara (Eds.), *Proceedings of The First International Conference on Advances in P2P Systems (AP2PS 2009)*, 198-202, IEEE Computer Society, 2009.
11. Mats Neovius, Fredrik Degerlund, Kaisa Sere, "*Inter-service Dependency in the Action System Formalism*". In: Gordon J. Pace, Gerardo Schneider (Eds.), *Third Workshop on Formal Languages and Analysis of Contract-Oriented Software, FLACOS'09* , 45 - 53, University of Oslo, Department of Informatics, 2009.
12. Petter Sandvik, Mats Neovius, "*A Further Look at the Distance-Availability Weighted Piece Selection Method: A BitTorrent Piece Selection Method for On-Demand Media Streaming*". *International Journal on Advances in Networks and Services* 3(3 & 4), 473-483, 2010.
13. Mats Neovius and Kaisa Sere. "*Mastering the Relevance of Subjective Information in Ubiquitous Computing*". Submitted to International Journal of Networked Computing and Advanced Information Management (IJNCM) Special issue on Social Informatics and COMputing (SICOM).

Part II

Original Publications

Paper I

An Abstract Model for Incentive-Enhanced Trust in P2P Networks

Mats Neovius

Originally published In proc. of Embedded and Ubiquitous Computing - EUC 2005 Workshops: UISW, NCUS, SecUbiq, USN, and TAUES, Nagasaki, Japan, December 6-9, 2005. , Lecture Notes in Computer Science vol. 3823, 602 - 611, Springer Berlin / Heidelberg, 2005.

An Abstract Model for Incentive-Enhanced Trust in P2P Networks

Mats Neovius

Department of Computer Science, Åbo Akademi University,
Lemminkäisenkatu 14, FIN-20520 Turku, Finland
mats@neovius.com

Abstract. Peer-to-Peer (P2P) networks have emerged as a prime research topic, partly due to the vast unexploited possibilities unrestricted distribution of the workload provides. The main hindrance for unrestricted exploitation of the P2P topology is, due to lack of security-related issues, the gullible attitude taken towards unknown agents. Therefore, the severity of the vulnerabilities caused by gullibility must be mended by other means, for example, by an effective incentive scheme encouraging agents to trustworthy behaviour. This paper presents an abstract model for incentive enhanced trust, to progressively assign the participating agents rights for accessing distributed resources, emphasising consistent behaviour. The model consists of a degrading formula, an illustrative incentive triangle and a best-effort distributed supervision model. Moreover, the same incentive model facilitates anticipation of future behaviour concerning any given agent founded on several distinct agents' opinion, suggesting that any knowledge concerning the counterpart is better than none.

Keywords: Peer-to-Peer networks, incentive, trustworthiness, anticipation.

1 Introduction

Reputation-based trust systems are widely studied and are probably the most realistic approach to anticipate future behaviour of an agent. Consequently, as in reality, there must exist a powerful incentive encouraging participants in a P2P network to credibly exchange information and act consistently benevolently. Thus, as mentioned by Kamvar, Schlosser and Garcia-Molina in [1], the identification must be a long-term user-specific, not relying on an externally assigned identity such as the IP address.

One way to encourage consistent behaviour is by assigning a covetous benefit to agents behaving benevolently. This gain should play the role of real-life money; it should be desirable and entitle to additional privileges. However, such an advantage attracts fraud in various forms. To describe the problems, it is essential to declare the basic frameworks and concepts which trust, in this case, is to be applied on.

1.1 Peer-to-Peer Networks

A P2P system implementing trust resembles inter-human communication in many ways. In a P2P network, all participating agents act as clients as well as servers and

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 602 – 611, 2005.

possess equal rights, which suggest to a self-policing structure. Therefore, the definition concerning P2P architecture to be used throughout this paper is as follows:

> A P2P architecture is a distributed network where each node grants other requesting nodes access to its shared resource(s). The resource(s) is/are accessible by any other participant on the network directly without intermediate servers.

Consequently, this paper views the participants in a P2P network as "members of a society", where an agent's actions are egocentrically determined by benefit. Moreover, a P2P system should be capable of handling any arbitrary agent's unexpected drop-off from the network at any given time, without the network suffering any loss of service [2]. This excludes implementation of a predefined structure, such as servers or pre-shared secrets.

Despite the exclusion of central units, we argue that deploying an incentive in an agent-centric P2P architecture, a structured overlay network is a necessity. This is motivated because it enables systematic knowledge lookup, efficient collaboration between the participating agents for maintaining the incentive and assignment of credit to the appropriate agent. This paper considers the overlay system organised as a Pastry Distributed Hash Table (DHT) architecture. The Pastry DHT system provides scalability, low network diameter and proximity selection [3, 4].

1.2 The Trust Metric

Trust is a social phenomenon and can only exist between two distinct matters of which at least one is capable of feeling. As such, all models of trust should be based on the same as the social trust, knowledge about the counterpart. This paper discusses unrestricted agent-centric trust and situations where it is assumed that the counterpart is behaving irrationally. Walsh and Sirer in [5] propose an object centric reputation scheme that is restricted to a specific kind of objects, in their case files. Such a system is however, unsuitable for agent-centric reputation evaluation because peers' behaviour vary.

Implementing trust to be processed in a microprocessor requires that it can be measured and thereby, compel assigning a value for the metric. In a binary formation, an agent is evaluated as either trustworthy (affirmative) or untrustworthy (negative). Eventually every assessment should fit the binary formation. Considering the perpetually changing environment and variety of levels demanded, binary formation is insufficient for comprehensive usage. Therefore, the trust metric is considered in this paper discrete, between 0 (none) and 1 (complete), with a sufficient amount of states. According to calculations made on the values, the trustor will assign the trustee-specific rights to access and/or exploit resource(s), as will the trustee select the provider.

Besides the value of the metric, it must be distinguishable on a per actor basis and thus, explicitly mapped to a unique ID, as humans are recognised by characteristics such as the voice, by sight etc. Consequently, we argue that a unique ID is a precondition for implementing trust of any kind between conditionally trustworthy matters.

1.3 Recognised Abuses in P2P Networks

The present gullible approach adapted by participants in a P2P network, and the limited possibilities to locate colluding agents, attracts abuses of many kinds. Concerning peer misbehaviours, three types are recognised: collude inflation, deflating and faking [6].

Collude inflation are situations where a conglomerate of agents collaborate by reporting positively about each other in order to achieve a higher trustworthiness value. The problem is present in centralised online auctions and in reality, because there is no way to verify the feedback's truthfulness and dignity. However, including only one report per agent such as in eBay.com [7] and degrading the information by time would hamper any colluding intentions.

Deflation is a situation where a set of agents defames another's reputation by reporting unsatisfactory behaviour concerning it. This is comparable to spreading rumours in reality. However, degrading of reports as for collusion, affects deflation equally and is a feasible countermeasure.

A faker is an agent that introduces itself as another agent (usually) possessing a higher reputation. This problem should be solved at the assignment of the ID or at the mapping of the feedbacks to an ID. However, this is out of the scope of the topic and this paper assumes that the ID's are unique and the feedbacks are authentic.

Besides the misbehaviours concerning reputations; annoyances such as "free-riding" and "tragedy of the commons" are widely acknowledged. Both are consequences of unfair exploitation and contribution respectively, of the commonly accessible resources and can be solved utilising the kind of incentive presented later in this paper.

2 Trust in an Open Environment

A trust relation can be of many forms; it can be one-to-one, many-to-one, or many-to-many [8]. Optimally, the relation is many-to-one, where the knowledge about the counterpart is based on a combination of several sources' experiences. However, a distinction of the knowledge credibility according to sources' trustworthiness is required. This paper considers a three-level hierarchy of knowledge sources: a personal opinion, trusted agents' opinions and a public opinion.

2.1 Personal, Trusted and Public Opinions

As when considering humans, trust between P2P-networked agents should equally count on the capability of distinguishing between trust derived from different matters and events. Deducing personal opinions based on personal experiences is essential. However, in some situations the observations cannot cover adequate knowledge and relying on others' judgements is necessary. The trusted agents' opinions are "advises" and acquired gathering information by inquiring friendly sources. A public opinion is one reflecting the majority's opinion concerning the matter. Hence, the personal opinion is a concern that is alterable only by the possessing agent. Moreover, each agent should contribute in providing and maintaining a public opinion and collaborate with personally trusted agents to enforce understanding about the counterpart, which

is/are considered advises and trusted conditionally. Thereby, the agents form sets of reciprocal trusted conglomerates.

The public opinion does not alone solve any of the problems mentioned. It should be considered as we consider, for example, reviews at epinions.com [9]. Therefore, the public opinion can, at most, mend the uncertainty left by the trusted and personal opinion. However, uncertainty should have primary influence on the decision concerning selecting the agent to process the event. This is motivated by the threat of a colluding set of malicious agents collaborating in building a benevolent public opinion.

Because of the anonymity and egocentric behaviour, it is justified that the personal opinion has greater influence on the final outcome than the trusted agents and public opinions [10]. Consequently, a hierarchy of credibility is formed where the personal opinion mends with the trusted and only then with the public opinion. This hierarchy severely hampers the effect of collusion and deflation. However, the different levels of opinions must not result in conflicts though possibly indicating an opposite outcome, and a method of achieving a consensus is needed. This consensus should handle situations such as, for example, when the public opinion suggests negative assessment while the trusted and personal opinion suggest to affirmative with some uncertainty. In addition, a consensus method of the opinions adjusts the personal opinion to the trusted and public opinions and reduces "obstinacy". The consensus of the different opinions results in a situation equal to inter-human interaction, i.e. that a maliciously behaving agent is capable of taking advantage of the conglomerate of reciprocally trusted agents' benevolence only a finite amount of times.

2.2 Feedback Formation and Distribution

Trust relying on a public opinion in P2P networks is motivated because no single entity can have accurate information about all others' conducts. Initially, no data concerning the counterpart exist, suggesting that reputation has to be built from some state. The state of no reputation, and thus the initial state, is considered in this paper as the state of uncertainty; because modelling trust in dynamic networks cannot allow confusion between "don't trust" and "don't know" [11].

A feedback is the generated data concerning the provider of the resource(s) after an event. The generated data is stored locally and submitted to the supervising agent including the ID of the counterpart, a timestamp, the feedback score and the ID of the reporter [12]. Additional application-specific data can be added. Including IDs in the feedback provides a possibility to identify and verify the transaction. In addition, the agents should monitor their reputation and when disagreeing on an evaluation, change the personal opinion about the reporting source accordingly. The timestamp enables utilisation of a degrading formula, with the justification that attitudes can change over time. The feedback itself is graded with a triplet of values; *belief* (b), *disbelief* (d) and *uncertainty* (u). As discussed in section 1.2 and because the metrics are in contradiction and complete equal 1, their sum must equal 1.

Considering the definition of a P2P network, the feedbacks must be stored on the connected (live) agents. As a countermeasure for colluding inflation, the agents supervising feedbacks concerning any given agent should perpetually change. Moreover, the agent the feedbacks concern should not be included in the lookup chain

of locating the supervising agent. Therefore, the feedback supervising agents must be known by all participants all the time. Enabling this in a system utilising Pastry DHT is possible by having the trust supervising agents' IDs dynamically assigned by a hard-coded function in the application. This requires the DHT to assign the IDs dynamically on a per-session basis as a countermeasure for colluding alternation. However, the need of a unique static ID for each participant compels usage of two interconnected DHTs, each consisting of x tier to maintain scalability. In such a system, one layer provides the static *nodeID* while the other layer accounts for proximity selection, lookups and the feedback, being dynamically assigned, hereafter denoted *sessionID* (*sID*). This way the needs of a static unique ID and the requirements for countermeasures are satisfied.

Requiring any reporter to file the feedback to, for example, two closest supervising agents of its own *sID*, would provide data redundancy. That is, if *sID* $c < d < e < f$, the agent with *sID e* files reputation regarding *sID c* to *sID d* and *sID f*. A possible recovery can be conducted by a logical expression, where peer *g*, *h*, *i*, *j* and *k* represents adjacent supervisors for *x*, according to the distribution. *i*'s stored data can be retrieved by $data \in h \vee j \wedge \neg g \wedge \neg k$. In other words, if *sID i* fails, its data can be recovered by summarising all data that *sID h* or *sID j* store and that are not stored by *sID g*, nor by *sID k*.

Moreover, the redundancy provides a way for a newly assigned supervising agent to verify the passed feedbacks. In addition, such *sID* data passing provides means for semi-symmetrical distribution. Consequently, in order for colluding inflation to succeed, the malevolent agent should cooperate with the majority of the involved dynamically changing supervising agents. The feedbacks reported to the supervising agents are the values resulting in the public opinion that is a sum, calculated by a subjective logic, for example, the one presented in [13], of all feedbacks from a set of interactions with the agent(s) concerned.

3 The Incentive

In reality an incentive is very simple. It is usually money, fame or some other covetous benefit that good performance entitles to. However, distribution of the beneficial is complex. The incentive to be deployed for usage in computerised communication must be based on the idea of giving benefit to the active and benevolent agents and reducing the value of the beneficial as a consequence of unsatisfactory performance. As a result, there must exist a *carrot* as well as a *stick*. In order to increase the anticipations truthfulness, experiences should degrade according to time.

3.1 A Degrading Formula for Trust

Philosophically, trust can never be absolute [14]. The core idea of this is the fact that even a friend, considered as trustworthy, can fail the expectations; respectively can an untrustworthy agent behave benevolently. To meet these challenges, a degrading formula must weaken the weighs of the feedbacks based on time and sociality. This is necessary in order to give less social agents equal possibilities; weakening recent

experiences less. Whitby, Jøsang and Indulska [15] proposed a formula without the sociality factor, however, including it in the same formula is easy, resulting in formula 1.

$$p_{Z,t_R}^{X,t} = \lambda^{t-t_R} \gamma^{(l-k)} p_{Z,t_R}^{X} \quad (1)$$

In formula 1, $p_{Z,t_R}^{X,t}$ is agent X's rating of agent Z at time t_R, t being the current time. In other words, an event occurred at time t_R where agent X rated agent Z, the current time being t. $0 < \lambda < 1$ is the longevity factor degrading the rating according to time. The γ^{l-k} represents the ordering of the feedback by occurrence, *l* being the selection's size and *k* the position number where the most recent is *l*, degrading according to sociality and being $0 < \gamma < 1$.

The formula should be applied upon the *belief* and *disbelief* values in personal opinions' every experience. Because the sum of the metrics is 1, uncertainty equals 1-*b*-*d*. In addition, formula 1 sociality factor covers the claim that complete trust or distrust cannot exist, and is a countermeasure for key-space depletion, dropping agents with uncertainty exceeding some predefined threshold value. The values assigned for λ and γ are subject to the application and the environment. The γ value should adjust to the frequency of attitude changes; the lower value, the heavier weight on recent events. λ depends on the frequency of transactions conducted with the counterpart. γ and λ combination reacts to changes in attitude and allow the agent to adapt to the environment. Moreover, the degrading formula is forgivable and will grant the maliciously behaving agent a new chance, after a given time, depending on the longevity factor, of acquiring favouring among the reciprocal conglomerate it tried to fool.

3.2 Calculating with the Metrics

Calculating and enforcing the accuracy of the metrics is essential in order to reach the decision. Figure 1 illustrates a situation where two trusted agents, Bob and Claire, contribute in enforcing Alice's anticipation concerning the target, David.

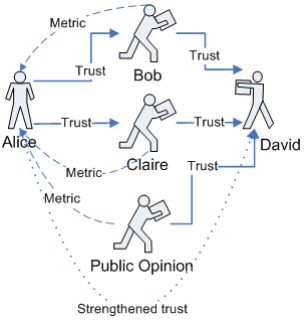

Fig. 1. Trust combination

The trusted agents participating in the evaluation should contribute with their personal opinions to the requesting entity, without enforcing their understanding by querying further or redirecting. This is motivated because Alice trusts Bob and Claire, not a fourth party, to evaluate David. A situation alike the one in Figure 1 compels a consensus to be achieved between Bob's and Claire's metrics. Bob's and Claire's consensus will eventually be combined with Alice's personal opinion, and finally patched by the public opinion, resulting in the final opinion.

The calculation merging the participating agents' degraded metrics is based on probability calculations and can be performed according to formula 2 illustrated below, originally proposed in [16].

$$\begin{aligned} belief_N^M &= (b^N * u^M + b^M * u^N)/(u^M + u^N - u^M * u^N) \\ disbelief_N^M &= (d^N * u^M + d^M * u^N)/(u^M + u^N - u^M * u^N) \\ uncertainty_N^M &= (u^N * u^M)/(u^M + u^N - u^M * u^N) \end{aligned} \tag{2}$$

M and *N* are any agents which personal opinion metrics are to be merged; in this case Bob and Claire. If several agents contribute, the merging is done between any two agents or sets of agents at the same level of the consensus process. Eventually the consensuses will reach such magnitudes that it represents the understanding of the underlying group.

The final patching of the uncertainty for the expected outcome utilising the public opinion should be performed after applying the metrics from the trusted agents. This can take place utilising, for example, the following formulas.

$$belief_{public}^{calculated} = belief^{calculated} + uncertainty^{calculated} * belief^{public} \tag{3}$$

$$disbelief_{public}^{calculated} = disbelief^{calculated} + uncertainty^{calculated} * disbelief^{public} \tag{4}$$

In these formulas, *calculated* denotes the degraded trustworthiness of the levels higher in the hierarchy, acquired by formula 2 and 1. Mending this calculated opinion with the public opinion that does not recognise uncertainty, forms an opinion correlating to the expected outcome based on the available knowledge.

Utilising these methods, the trust metric fits the triangle illustrated in Figure 2, when uncertainty is included and the anticipation of forthcoming behaviour is possible. Thus, all possible providers of the requested service can be compared and the most suitable chosen.

3.3 An Incentive View

In every incentive method, the inducement must be such that the users cannot gain from reinitiating with a new identity [1]. Hence, we argue that the initial state must be equal to or worse than the state of untrustworthy, with the justification that any knowledge to base anticipation on reducing the risk of misjudgement is better than none. This results in the idea that the state of disbelief is preferred to the initial state, countermeasuring whitewashing.

This paper considers the initial state as the state of uncertainty, a state where no anticipation about future behaviour based on reputation is possible. At the same time,

the state of uncertainty indicates that the ID is available for any requesting newcomer. The incentive triangle, derived from the opinion triangle in [16], illustrated in Figure 2, summarises these ideas.

The triangle should be interpreted so that each vertex represents completeness. Therefore, the trustworthiness of any agent consisting of three metrics is representable by one point in the triangle. The median starting at each vertex is the grading of the different values, where belief is represented by Q, disbelief by R and uncertainty by P. The dot represents an example (personal opinion), with belief (Q) 0.25, disbelief (R) 0.65, uncertainty (P) 0.1. E(x) presents the mending (expectation), illustrated in formula 3 and 4, where the personal opinion's uncertainty is mended by the public opinion, whose value is represented by the dotted line a_x. In Figure 2, this starts at uncertainty, ending at *belief* = 0.6 and thus *disbelief* = 0.4.

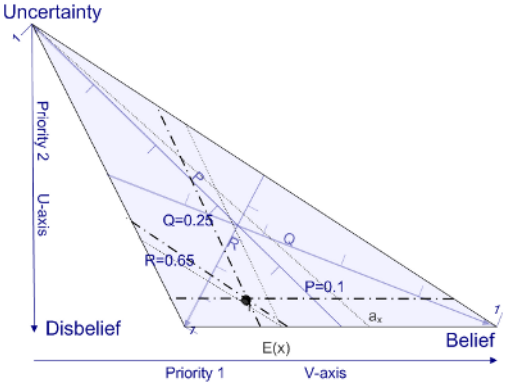

Fig. 2. Incentive triangle

When calculating the expectations value, the final value is required to be either affirmative or negative and thus uncertainty must equal 0. Uncertainty is reduced to equal 0 by applying the public opinion on formula 3 and 4, resulting in the removal of the uncertainty metric. The degrading formula 1 affects the opinion in the way that it moves towards uncertainty on the axis with the original relation between trust and distrust. Moreover, the triangle recognises two priorities, which are determined by trust qualities and thus purpose specific.

In this specific view, a newcomer is not assigned any profit, which should be the best countermeasure for avoiding an agent with bad reputation to reinitiate its trust relation in form of signing in with a different ID. This implies that the participants are encouraged to consistently act using the same identity every time.

The presented ideas maintain a balance between capability to operate and actual trustworthiness. If some agent is incapable of fulfilling the placed expectations, its trustworthiness will suffer among the expecting agents. Consequently, the network has reacted to this successfully and the trustworthiness/capability balance is maintained.

4 Conclusion

Combining the models presented in this paper reduces the presented problems' severity. Colluding inflation can occur a finite amount of times per conglomerate of reciprocally trustworthy agents because of the influence of the personal opinions. Deflation compromises the public opinion but the target maintains its ability to operate due to the personal opinions and will recover because of the degrading formula. Issuing countermeasures for faking is very difficult, if not impossible, without pre-shared secrets or intermediate authenticating servers. The "free-rider" and the "tragedy of the commons" problems are solved by a *carrot - stick* relation and utilisation of the personal and public opinion. In addition, the presented incentive reacts to changes in attitude and provides a possibility for malevolent/passive behaviour to change without re-identification.

The problems remaining are the evidence concerning a feedback and the assigning of a unique ID. These issues are of different character and we cannot see the way these could be solved utilising an incentive. Moreover, credentials are excluded from this paper, but being an extension of trust relationships, they are an essential part of trust in reality.

Any accurate simulations to enforce the claims in this paper are difficult to make because the contribution is in anticipation of the irrational. Simulations can thereby not reach greater accuracy than having a static value to calculate irrationality from, which is superficial. The reason is that this would imply simulating human behaviour, but since the human society is functional, creating a similar environment for computerised communication should be the objective. This paper has provided some ideas in order to reach this objective from the point of view that nature has evolved the ultimate trust formation scheme.

References

1. Spandar D. Kamvar, Mario T. Schlosser, Hector Garcia-Molina (2003). "The EigenTrust algorithm for reputation management in P2P networks". In Proceedings of the Twelfth International World Wide Web Conference, May, 2003.
2. Rydiger Schollmeier (2002). "A Definition of Peer-to-Peer Networking for the Classification of Peer-to-Peer Architectures and Applications". Proceedings of the First International Conference on Peer-to-Peer Computing (P2P'01).
3. Dan S. Wallach (2002). "A Survey of Peer-to-Peer Security Issues". International Symposium on Software Security (Tokyo, Japan), November 2002.
4. Miguel Castro, Peter Druschel, Y. Charlie Hu, Anthony Rowstron (2002). "Exploiting network proximity in peer-to-peer overlay networks". Technical Report MSR-TR-2002-82 (2002).
5. Kevin Walsh, Emin Gun Sirer (2005). "Thwarting P2P Pollution Using Object Reputation". Cornell University, Computer Science Department Technical Report TR2005-1980.
6. YangBin Tang, HuaiMin Wang, Wen Dou (2004) "Trust Based Incentive in P2P Network". Proceedings of the IEEE International Conference on E-Commerce Technology for Dynamic E-Business (CEC-East'04).

7. eBay.com. “Understanding feedback scores”. URL: http://pages.ebay.com/help/feedback/feedback-scores.html
8. Tyrone Grandison, Morris Sloman (2000). “A survey of trust in internet applications”. 4th Quarter 2000 issue of IEEE Communications Surveys & Tutorials.
9. Epinions.com. http://www.epinions.com
10. Vinny Cahill, Elizabeth Gray, Jean-Marc Seigneur, Christian D. Jensen, Yong Chen, Brian Shand, Nathan Dimmock, Andy Twigg, Jean Bacon, Colin English, Waleed Wagealla, Sotirios Terzis, Paddy Nixon, Giovanna di Marzo Serugendo, Ciarán Bryce, Marco Carbone, Karl Krukow, Mogens Nielsen (2003). “Using trust for secure collaboration in uncertain environments”. IEEE pervasive computing, volume 2, number 3, July – September 2003, page 52 – 61.
11. Marco Carbone, Mogens Nielsen, Vladimiro Sassone (2003). “A formal model for trust in dynamic networks”. BRICS Report RS-03-4, 2003.
12. Li Xiong, Ling Liu. “PeerTrust: Supporting reputation-based trust for peer-to-peer electronic communities”. IEEE Transactions on knowledge and data engineering, vol. 16, no. 7, July 2004.
13. Audun Jøsang (2001). “A Logic for Uncertain Probabilities”. International Journal of uncertainty, Fuzziness and Knowledge-Based Systems. 9(3), pp.279-311, June 2001.
14. Martin Hollis (1998). “Trust within reason”. Cambridge, United Kingdom, Cambridge university press.
15. Andrew Whitby, Audun Jøsang, Jadwiga Indulska (2004). “Filtering Out Unfair Ratings in Bayesian Reputation Systems”. In the Proceedings of the Workshop on Trust in Agent Societies, at the Third International Joint Conference on Autonomous Agents \& Multi Agent Systems (AAMAS2004), New York, July 2004.
16. Audun Jøsang (1997). “Artificial Reasoning with Subjective Logic”. In Proceedings of the Second Australian Workshop on Commonsense Reasoning, 1997.

Paper II

A Design Framework for Wireless Sensor Networks

Mats Neovius and Lu Yan

Originally published In proc. of IFIP WCC 2006, Ad Hoc networking track, Santiago de Chile, Chile, August 20-25 pp. 119-127. ISBN: 0-387-34635-X

A Design Framework for Wireless Sensor Networks

Mats Neovius, Lu Yan
Åbo Akademi University, Department of Information Technologies,
Lemminkäisenkatu 14, FIN-20520 Turku Finland.
{mneovius, lyan}@abo.fi

Abstract. Wireless sensor networks (sensornets) are wirelessly communicating smart gadgets with the capability of sensing the environment. With the immense applicability of sensornets, there is an increasing need of a general organisational and architectural development framework for sensornet systems. This paper outlines an abstract framework for modelling responsibilities and tasks to sets of nodes according to their vocation. These guidelines are presented with the intension to ease reasoning about a sensornet as a system, and its applications.

1. Introduction

The amount of research conducted regarding wireless sensor networks (sensornets) is emerging. The concept of sensornets envisions a new ambitious paradigm of computing, brought forth by Weiser in 1991 [1], usually referred to as ubiquitous or pervasive computing.

Large scale sensornets are complex and challenging environments in which to develop software. The applicable areas for ubiquitous sensors providing raw unprocessed data about the environment are vast. Moreover, sensornets constitute several Internet-era challenges, making them interesting for the research community as well as for industry.

Typically, a sensornet comprises a set of energy constraint nodes which, in addition to amorphous Ad Hoc networks, relies on collaboration with each other. The main advantage, from a research point of view, compared to more efficient computing units is that the sensornet node has only a limited number of reasonably executable tasks, which it is designed for.

The future potential of sensornets is immense. Sensornets provide a sensible transition towards ubiquity and pervasiveness, which might very well be the next step in the development of computing gadgets. If so, sensornets might trigger a new "era" in computing, like the one entered when the computers shrank to desktop size.

Only human imagination is the limit for what sensornets ubiquity can assist in and/or do for us when brought around and integrated to our environment and daily life. In order for this to happen, the units must be miniaturised. In minimised gadgets, the energy supply constitutes a significant portion of the total size. Hence, there are two ways to proceed; decreasing either energy consumption or battery size.

Many ideas and implementations utilising the ubiquity of a sensornet have already been presented, one of the most well known is the smart home with the example refrigerator automatically composing the shopping list [2]. Technically, this has been done and is available. The questions arising today address what humans are willing to learn, use and long for. Consumers have comprised as the test bed for the past era of computing development and a kind of technical saturation might come up. Consequently, a transition towards ubiquity, where the system filters relevant from irrelevant data, and assist in decision making is likely to be about.

The sensornet could thereby be viewed as a wirelessly inter-communicating encapsulated environment harvesting raw data with its sensors. The sensors extract measurements from its surroundings, that might be further refined in others, for that specific task dedicated units. The sensornet, as an architecture, ends where the data is passed to gadgets not fulfilling the criterions of a wireless sensor. Because the encapsulated nature and limited functionality, it is also attractive to make an effort to reuse code or parts of it.

Research regarding sensornets is often interdisciplinary, usually concerning at least the areas of computer science and electrical engineering. There are plenty of unsolved issues in various fields of study within the area. From a software point of view, there is a demand for novel ideas in areas concerning human-computer interaction, energy-saving, optimisation, self-organisation, information composition, query propagation and miniaturisation to mention a few. Consequently, sensornets assert the extreme of many problems in computing related disciplines.

We argue that in order for achieving a breakthrough in sensornets, a consensus regarding a general system framework for declaring which computations are performed on which parts of the network is necessary. If done, the network could apply the most suitable existing method for each situation.

The organisation of this paper is as follows. In section 2, we discuss the fundamental building blocks, identifying sensornets, from a perspective of hardware, functionality and middleware. The proposed system design framework is presented in section 3. Finally, we conclude the paper in section 4.

2. Fundamental building blocks of sensornets

The amount of separate building blocks of any system depends on the level abstraction it is viewed at. In this paper, we take a high-level of abstraction in order

to keep the ideas scalable and as general as possible, to fit sensornets from small stationary static environments to vast dynamic mobile networks.

A sensornet can be viewed as an encapsulated end-to-end mini-world with limited energy. The nodes energy capacity varies within the network. Moreover, for a sensornet to supply any service, it must have an interface for external data consumption. If the system provides means for bidirectional data flow, an overlay structure to organise query propagation is required.

The aim of the system is providing a method to obtain raw data and fuse it with appropriate context. Because the sensornet is a raw data provider for a service, it must address all the different parts; interface, propagation, data extraction and so forth. Moreover, each node must be able to function independently and collaborating when suitable. Thereby, dynamicity is a core issue to address. The highest priority for the system is to reply any proper query origin and deliver the requested service to the inquirer.

2.1 Hardware blocks in sensornets

Unfortunately, there is no commonly agreed definition for what a wireless sensor is, and what it is not. In order for providing a system framework for the sensornet to be applied on, an explicit definition is demanded. Deducing a definition from the meanings of the words wireless, sensor and network seems right, [3] described the concept as a simple equation which is supported by [4]:

"Sensing + CPU + Radio = Thousands of potential applications" [3]

[5] adds to this equation a power unit. However, this definition covers, for example, a laptop with WLAN capability that adjusts its display contrast to the environments luminosity, which was not the original idea of the equation.

With the compelling need of a definition, we agree on the equation, except for the term "radio" which we would like to replace with "wireless transceiver". The reason is that wirelessness does not necessarily equal radio-transmission. Moreover, we would like to add that a wireless sensor is usually a stand-alone small-scale device. Hence, this is the definition to be used throughout this paper.

The constituting compulsory blocks are thereby the clear-cut power unit, sensor(s), CPU (and consequently some memory) and the transceiver(s). Sensing capabilities are restricted by energy consumption and the physical size. The CPU power is restricted by the energy source capacity and should respond to the given sensor's needs, e.g. measuring temperature do not require much CPU power. The transmitter is the single device usually consuming the majority of the available energy. Consequently, energy efficient routing in self organising mesh networks attracts researchers focus. All of these units are connected to each other on a motherboard-like circuit, usually referred to as the *mote*.

2.2. Functional classification of sensornets

As described earlier, the sensors sense the environment and produce raw data, for example, "+20°C". Naturally, the amount of information this data provide without the context of location is limited. The context is added by another sensor connected to the same mote or by data composition[1] with data from another device. Regardless of the extent the data is composed of and refined to, it must finally be representable and becomes relevant only when it is sufficient enough to influence a decision. However, still at this era of ubiquity, the decision is often made by a human, on the top of the system hierarchy.

As stated, data without context destitute information and distinct raw data seldom have context. Considering sensornets, the context of the specific data becomes crucial. Any unit composing the data possesses additional knowledge that combined increases the amount of information. For example, in a simplified case, three distinct measurements are composed to provide relevance, temperature, location and time that might origin from distinct nodes. Unless this device is the gateway, there is a system hierarchy consisting of at least two levels.

In order to efficiently utilise available energy, moderate sized sensornets routing employs multi-hop protocols [6, 7, 8]. In many ways, the protocols resemble ideas used in decentralised mesh networks. The network is often fragmented and "cluster heads" are appointed [9]. Consequently, the framework must handle systems that are hierarchical to an undefined depth as well as flat networks, in order to preserve scalability and generality.

If the sensornet nodes are heterogeneous, with nodes dedicated for a specific tasks such as communication (more energy), locating (for example, GPS), their special capabilities should be taken into account when initialising the network. Thereby, we classify nodes in a sensornet as follows:

1) Sensing node(s)
2) En route node(s)
3) Gateway node(s)

The sensing nodes are the "bottommost" nodes in the system hierarchy, the ones sensing the environment. The en route nodes are devices that act as cluster heads or forwarders of the data between its endpoints, and possibly aggregate [2] or/and compose the raw data. The obligation of acting as an en route node is, due to energy capacity, traffic load and network lifetime, in some cases altered between nodes according to the routing method. Consequently, the nodes classified in class 1 and 2 should vary for efficient utilisation of network resources. The gateway node(s)[3] is responsible for the "topmost" level of a sensornet and according to the definition, the upper boundary. This node acts as the interface towards an external data consumer, for example, the Internet.

[1] Composition: Two distinct parts of data combined to be one.
[2] Aggregation: Two distinct parts of data embedded with their key characteristics into one packet in order to save energy consumed in transmission.
[3] Gateway node: Considered written singular though possibly plural occurrence.

The gateway is the interface to the outside. Any node can belong to one or more classes at the same time. In special cases, one node can constitute in all taxonomy, meaning that the gateway's underlying network size is one.

2.3. Middleware and components

Middleware technologies in a broad sense, which covers operating systems and virtual machines, query processing, data composition and aggregation, resource awareness and energy harvesting, overlay routing and communication management, etc., have the potential to ease and accelerate software development in sensornet environments by offering simplified application-level views that abstract over factors such as the above.

As a supporting example, as well as prevailing paradigm, lots of experimental sensornets today run on top of TinyOS [10] and TinyDB [11]. The first, TinyOS, is an open-source operating system specially trimmed for sensornets. It features a component-based architecture which enables rapid prototyping and implementing sensornet applications via providing higher-level programming abstractions. The latter, TinyDB, is a query processing system for extracting information from sensornets made from sensors running TinyOS. It features a SQL-like query interface technology which alleviates the complex of writing low-level C codes and supports traditional database queries with auxiliary sensornet parameters.

3. The design framework

A system design framework for sensornets is longed for, as Culler et. al. conclude: "We contend that the main obstacle limiting progress in sensornet work is the lack of an architecture. A sensor network architecture would factor out the key functionalities required by applications and compose them in a coherent structure, while allowing innovative technologies and applications to evolve independently" [12]. [5] describes the sensor networks protocol stack as 2-dimensional with six communication layers and three management planes.

We agree with both, but in addition tackle the issue from a "horizontal" view of node vocation, making the framework 3-dimensional. The 3-dimensionality is necessary in order to give the sensornet an overview of the system's status and adapt to it. Adjustment to prevailing situation is made by altering the routing method, changing functionality between reactive, proactive and hybrid protocols or by any other modification.

The strength is the utilisation of the core quality of each node, "because any specific context can often be provided by a variety of different types of sensors and used by different applications" [13]. We describe a general system framework for implementation on any sensornet platform that meets with the constraints described in section 2.

3.1. The layers

To factor out the key functionalities, a viable sensornet system design framework must partition the model to a structure with “black-boxes”. This way the developer needs to know only the task and the interface of the box in order to develop a replacement, use, test or evaluate it. “To become a reusable asset, it is not enough to start with a monolithic design of a complete solution and then partition it into fragments Instead, descriptions have to be carefully generalised to allow for reuse in a sufficient number of different contexts” [14]. Thus, developers are able to tune the sensornet upon the system framework according to their preferences.

As described in section 2.2 and 3, the framework have *n* horizontal and at least 3 vertical layers. Figure 1, deduced from [15], illustrates the vertical layers and horizontal node classes combined with the diagonal execution ellipse. [5] motivated the 2-dimensionality on each sensor, which is considered.

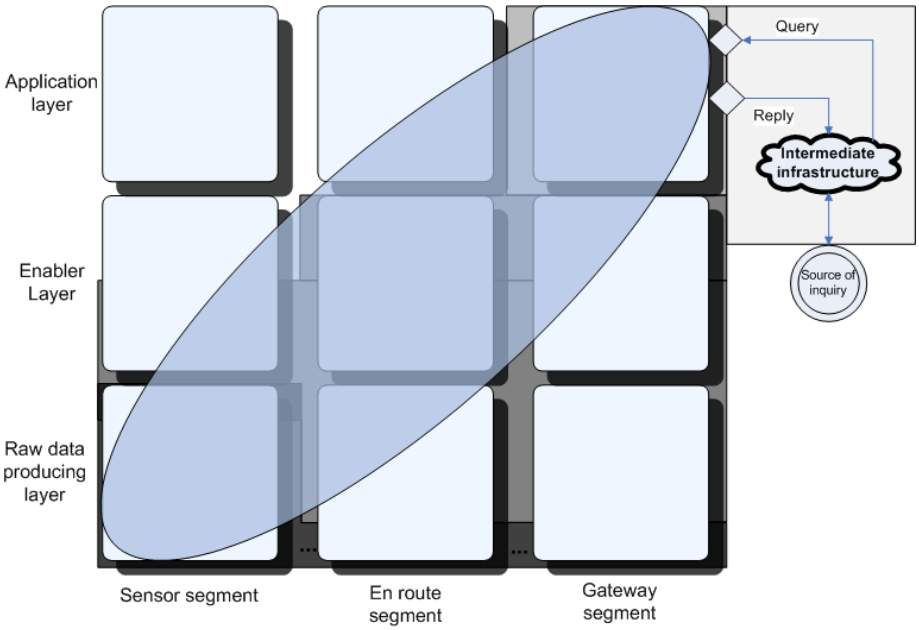

Figure 1. The sensornet system framework

The grey-shaded angular areas illustrate the main responsibility for the sensors belonging to them, where the dark grey area constitutes the sensing nodes, the grey the en route nodes and the light-grey the gateway node. Moreover, Figure 1 should be interpreted so that each item is considered belonging primarily on the “layer” and secondarily to the “segment”. The unified sensing system model presented in [16] supports the idea, layers and tasks meet in the ellipse.

A contribution in this framework is that all sensors do not necessarily provide data needed for replying a query, nor does all function as en route nodes. Consequently, the en route nodes can decide based on the query whether their

underlying sensing nodes can provide relevant information and thereby, decide to forward or not.

Moreover, the framework in Figure 1 could, if needed, illustrate a subset of a complete sensornet system and there might potentially be several such models in parallel interconnected by, for example, the Internet. As an example, one subset might concern the heating adjustments in a building whilst another is responsible for logging the temperature near by. Combining the data from these two completely distinct systems refines the information.

The ellipse describes issues the system framework emphasises on the different classes. Vaidya et al. [16] present a strict hierarchy for sensor management and configuration used for solving a tracking problem. The model is applicable with minor modifications for different applications and supports the ellipse. Huebesher and McCann describe a middleware's context provision, which is a three level hierarchy [13]. Additional service providers and refiners could easily be added in this scenario supporting the ellipse of node vocation.

3.2. Query propagation and reply composition

Query propagation and reply composition are the things affecting QoS (quality of service) and quality of context the most. Consequently, the system robustness is preserved during these phases. In addition to providing QoS, propagation and composition should preserve energy by merging into packets payloads, reducing transmission. According to studies, the ratio of sending one bit compared to one CPU-instruction is in WINS NG 2.0 nodes around 1 to 1400 [17] and usually considered to be approximately 1 to 1000. Hence, it is motivated to emphasise the critical parts affecting consumption of the scarce resources, the en route nodes.

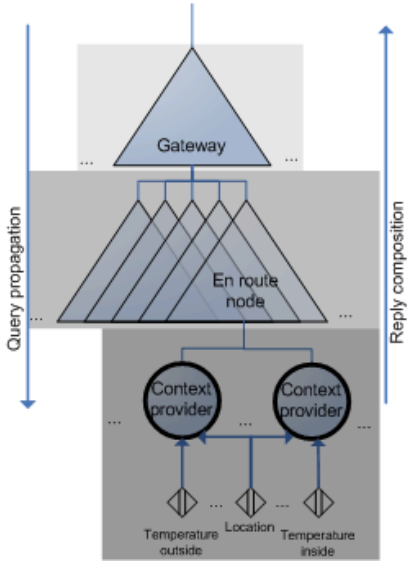

Figure 2. Data propagation / composition

Query propagation and reply composition are opposite to each others and can theoretically take place anywhere en route, see Figure 2. Fundamentally, the inquirer expects providing of announced service, whether it is a user or a layer above. The query must be properly propagated down the layers until replied or reaching the "bottom" and the raw data replied composed with context, providing relevance.

Figure 2 illustrates how data is propagated and composed in a 3-level hierarchical system. The context providers provide distinct raw data that is in the en route node composed to increase information. The gateway finally functions as the interface. Placing this figure diagonally on the framework provides an illustration of node vocation and executing tasks.

A reply for the query can also be processed at any node en route. This depends on the context-awareness method used. According to Chen and Kotz [18], two different kinds exist and they defined them as following:

> Active context awareness: an application automatically adapts to discovered context, by changing the application's behavior.
>
> Passive context awareness: an application presents the new or updated context to an interested user or makes the context persistent for the user to retrieve later.
> [18]

The similarity of these to reactive and proactive data passing modes in sensornets is evident. Recalling the examples mentioned in section 3.1, adopting the heating to temperature variations would be active context awareness whilst logging outside temperature is an example of passive.

An additional strength of our system framework is the possibility to differentiate between layers in the data forwarding hierarchy. The advantage is that different layers can adopt different operating modes. Consequently, dynamically adapting to application demands by implementing active or passive modes in a system can save energy.

4. Conclusions

We argue that today, the main task is to harvest as much information as possible. However, with the development and ubiquity of processing units, we anticipate an overwhelming magnitude of available information in the future. Thereby, the challenge will be to differentiate between "data" and "relevant data".

In this work we have presented a framework for systematic development of sensornet applications. The proposed framework is supported by numerous works and binds together the fundamental points in them. Its level of abstraction covers known demands and adapts to new situations. It eases reasoning and provides a method upon which to facilitate the development of new innovative applications in sensornets.

References

1. M. Weiser, The Computer for the Twenty-First Century, Scientific American, Sept., 1991.
2. A. C. Huang, B. C. Ling, S. Ponnekanti, A. Fox, Pervasive Computing: What is it Good for?, In Workshop on Mobile Data Management (MobiDE) in conjunction with ACM MobiCom, 1999.
3. J. Hill, System Architecture for Wireless Sensor Networks, Ph.D. Thesis, UC Berkeley, 2003.
4. A. Wadaa, S. Olariu, L. Wilson, M. Eltoweissy, K. Jones, Training a wireless sensor network, Mobile Networks and Applications, Volume 10, Issue 1-2, February 2005, Pages: 151 – 168.
5. I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, A survey on sensor networks, Communications Magazine, IEEE, Volume: 40, Issue: 8, Aug 2002, page(s): 102- 114, ISSN: 0163-6804.
6. A. Woo, T. Tong, D. Culler, Taming the Underlying Challenges of Reliable Multhop Routing in Sensor Networks, In Proc. ACM Conference on Embedded Networked Sensor Systems (SenSys'03), 2003.
7. W. Ye, J. Heidemann, D. Estrin, An Energy-Efficient MAC Protocol for Wireless Sensor Networks, In Proc. IEEE INFOCOM'02, 2002.
8. W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient Communication Protocols for Wireless Microsensor Networks, In Proc. IEEE HICSS'00, 2000.
9. Jamal N. Al-Karaki, Ahmed E. Kamal, Routing techniques in wireless sensor networks: a survey, IEEE Wireless Communications, ISSN: 1536-1284, Dec. 2004, Volume: 11, Issue: 6, page(s) 6- 28.
10. TinyOS: http://www.tinyos.net/ .
11. TinyDB: http://telegraph.cs.berkeley.edu/tinydb/ .
12. David Culler, Prabal Dutta, Cheng Tien Ee, Rodrigo Fonseca, Jonathan Hui, Philip Levis, Joseph Polastre, Scott Shenker, Ion Stoica, Gilman Tolle, Jerry Zhao, Towards a Sensor Network Architecture: Lowering the Waistline, Tenth Workshop on Hot Topics in Operating Systems (HotOS X), Eldorado Hotel, Santa Fe, NM, USA, June 12–15, 2005.
13. Markus C. Huebscher, Julie A. McCann, Adaptive middleware for context-aware applications in smart-homes, ACM International Conference Proceeding Series; Vol. 77 archive, Proceedings of the 2nd workshop on Middleware for pervasive and ad-hoc computing, Toronto, Ontario, Canada 2004, Pages: 111 – 116, ISBN:1-58113-951-9.
14. Clemens Szyperski, Dominik Gruntz, Stephan Murer, Component Software - Beyond Object-Oriented Programming, Second Edition Addison-Wesley / ACM Press, 2002. ISBN 0-201-74572-0.
15. Jurgen Ziegler, End-to-End Concepts Reference Model, Nokia, 2003.
16. D. Vaidya, J. Peng, L. Yang, J. W. Rozenblit, A Framework for Sensor Management in Wireless and Heterogeneous Sensor Network, ecbs, 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), 2005, pp. 155-162.
17. V. Raghunathan, C. Schurgers, Sung Park, M.B. Srivastava, Energy-aware wireless microsensor networks, Signal Processing Magazine, IEEE Volume 19, Issue 2, Mar 2002 Page(s):40 – 50. 2002.
18. Guanling Chen, David Kotz, A Survey of Context-Aware Mobile Computing Research, Department of Computer Science, Dartmouth College, Dartmouth Computer Science Technical Report TR2000-381.

Paper III

A Formal Model of Context-Awareness and Context-Dependency

Mats Neovius, Kaisa Sere, Lu Yan and Manoranjan Satpathy

Originally published in proceedings of Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06), 2006 pp. 177-185.

A Formal Model of Context-Awareness and Context-Dependency

Mats Neovius, Kaisa Sere, Lu Yan, M. Satpathy
Dept. of Information Technologies, Åbo Akademi University, FIN-20520 Turku, Finland
{Mats.Neovius, Kaisa.Sere, Lu.Yan, Mannu.Satpathy}@abo.fi

Abstract

The communication environment surrounding our daily experience is increasingly characterized by mobile devices that can exchange multimedia information and provide access to various services of complex nature. The trend is now clear that future consumer computing experience will be based on multiple pervasive communication devices and services, where navigability, context-sensitivity, adaptability and ubiquity are key characteristics. Several issues have been studied, models and methodologies proposed, and tools and systems implemented. However, we look at the foundation, where some of the most relevant issues probably are a formal model of context-awareness and context-dependency. In this paper, we discuss a formal foundation and software engineering techniques for mobile context-aware and context-dependent service derivation and application development, emphasizing the relationships between context and system.

1. Introduction

With more than two billions terminals in commercial operation world-wide, wireless and mobile technologies have facilitated in the first wave of pervasive communication systems and applications. This trend shows several aspects consistent in the evolution of computing including the increasing miniaturization of the computing units and an increasing emphasis of the role of communication between them. Significant research work has been done over recent years on these systems at several levels, from the lowest physical level to the highest information processing level. However, the latter is less developed than the research at the lower levels. For instance, we think that the most relevant issue for the future perspective of true ubiquitous computing, *context-aware and context-dependency* has not received justified attention in the research community.

The term *context* has been extensively studied since the early 1990s; it was mainly associated with the concept of *location*, but it is much richer than this; some works have categorized context into different aspects, such as computational, user, physical, spatial and temporal context [1, 2, 3, 4, 5, 6]. However, there is no consensus on the semantics of the word context in the literature. In order to reason about the concept, we interpret context as *a setting in which an event occurs*, and this construe, we believe, is suitable for the system software research.

In a previous work [7], a formal approach to context-aware mobile computing is described: we offer the context-aware action systems framework, which provides a systematic method for managing and processing context information, defined on a subset of the classical action systems [8]. Based on the essential notions and properties of this formalism, we applied this formalism in deriving context-aware services for mobile applications [9], and implemented in a wireless sensor network a smart context-aware kindergarten scenario where kids are supervised unobtrusively [10].

Issues that have been considered are both theoretical and practical: modeling the system requirement rigorously with formal approaches, deriving the software architecture from formal models, stepwise refinement of the specification, code generation, and verification vs. simulation. While all these research issues have been individually studied in an extensive way, their interaction within the final implementation raises new challenges, which constitutes the focus of this paper.

The remainder of the paper is organized as follows: after a short introduction to related work in section 2, a design framework for wireless sensor networks is presented in section 3. In section 4 we describe a formal model of context-awareness and context-dependency, and show the relationship between the model and software architectures. We discuss a case study on applying this model to software development process in section 5, and then conclude the paper in section 6.

Proceedings of the Fourth IEEE International Conference on Software Engineering and Formal Methods (SEFM'06)
0-7695-2678-0/06 $20.00 © 2006 IEEE

2. Related work

Several related works have noticed the importance of seeking a foundation of context-aware computing [22]. Roman *et al.* presented a formal treatment of context-awareness via extending the mobile UNITY with context handling part into context UNITY [23]. The context UNITY formalism is similar to our context-aware action systems formalism, but approaching from an agent-like view in modeling context-awareness and context-dependency.

Henricksen *et al.* showed a conceptual framework and software infrastructure that together address known software engineering challenges in context-aware computing applications [24]. The context model is built at the semantic level using the CML language [25], which can be categorized as an extension of the Object-Role Modeling in software engineering process.

UML approach to context models was presented by Hinze *et al.*, where UML diagrams are combined with discrete event systems to facilitate the development of mobile context-aware systems [26]. Due to the limitation of UML, which lacks a rigorous mathematic foundation, this approach can be deemed as a semi-formal one. The similar UML-like approach can be found [27], where a simulation-based paradigm was presented. Besides general aspect of context, fragment aspects of context, such as ontology [28], rational [29], middleware [30], trust [31] were also considered.

3. Wireless sensor networks

Wireless sensor networks provide perfect platforms to study context-aware and context-dependent systems on. Wireless sensor networks have been an area of active research since the early 1990s [11], accelerated by the advancement of wireless networking and the development of sensors. Only recently, wireless sensor networks have moved from academic research concepts to commercially available products, increasing production quantities.

Although significant research work has been undertaken, most of the research is still very application specific, with security and environmental applications dominating [12]. However, it is likely that more generic and comprehensive approach is required, where true system level problems in wireless sensor networks and their applications can be studied. With such a perspective, we deduced **Figure 1** from the design framework for wireless sensor networks proposed in [13].

In this framework, we have distinguished between *context-provider* (CP) and *context-utilizer*; the former is the reactive part which detects the surroundings and acquires the context, and latter is the proactive part which interprets and responds to the context. The interaction between the context-provider and context-utilizer constitute a complete context-aware and *context-dependent* system (CD). A context-dependent part of the system depends on a context-provider to supply the metrics for fulfilling its declared service.

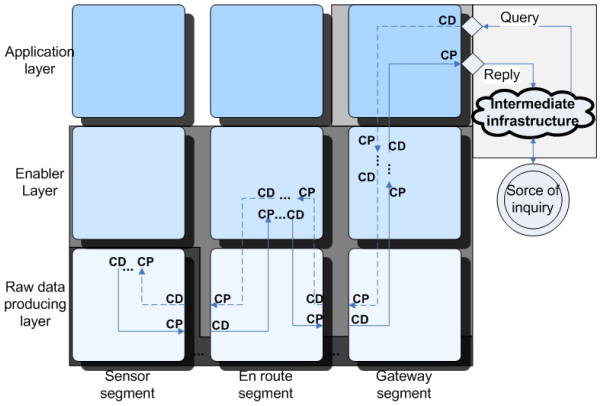

Figure 1. The sensornet system framework

Because the possibly bi-directional communication and the impossibility of restricting context to be a sensor reading, all nodes can potentially act as context-providers as well as context-utilizers. The roles are dependent on whether the data is propagating (an inquiry) or composing (a reply).

4. Formalizing context-awareness and context-dependency

We start by giving a brief overview of the action system formalism and then present how we model context-awareness and context-dependency within this formalism. By mapping the formal model back to the software architecture of wireless sensor networks, we show some realistic implementations of this model on system software research.

4.1. Action systems

The action systems formalism is based on Dijkstra's language of guarded commands [14]. This language includes assignment, sequential composition, conditional choice, and iteration.

4.1.1. Actions

An *action* is a guarded command, i.e. a construct of the form $g \rightarrow S$, where g is a predicate, the *guard*, and S is a program statement, the *body*. An action is

said to be *enabled* when its guard is evaluated to *true*. If an action does not change the program state it is called a *stuttering* action

The body S of an action is defined as follows:

$$S ::= \text{abort} \mid \text{skip} \mid x := e \mid \{x := x' \mid R\} \mid$$
$$\text{if } g \text{ then } S_1 \text{ else } S_2 \text{ fi} \mid S_1; S_2$$

Here x is a list of attributes; e is a corresponding list of expressions, x' is a list of variables standing for unknown values, and R is a relation specified in terms of x and x'. Intuitively, *skip* is an stuttering action, $x:=e$ is a multiple assignment, *if g then* S_1 *else* S_2 *fi* is the conditional composition of two statements, and $S_1;S_2$ is the sequential composition of two statements. The action *abort* always fails and is used to model disallowed behaviors. Given a relation $R(x,x')$ and a list of attributes x, we denote by $\{x:=x' \mid R\}$ the *non-deterministic assignment* of some value $x' \in R.x$ to x (the effect is the same as *abort*, if $R.x = \phi$).

The semantics of the actions language has been defined in terms of weakest preconditions in a standard way [14]. Thus, for any predicate p, we define:

$wp(\text{abort}, p) = \text{false}$
$wp(\text{skip}, p) = p$
$wp(x := e, p) = p[x := e]$
$wp(\{x := x' \mid R\}, p) = \forall x' \in R.x \cdot p[x := x']$
$wp(S_1; S_2, p) = wp(S_1, wp(S_2, p))$
$wp(\text{if } g \text{ then } S_1 \text{ else } S_2 \text{ fi}, p) = \text{if } g \text{ then } wp(S_1, p)$
$\text{else } wp(S_2, p) \text{ fi}$

where $p[x:=e]$ stands for the result of substituting all the free occurrences of the attributes x in the predicate p.

4.1.2. An action's building blocks

An *action system* is a construct of the form:

$$\begin{aligned} A = \mid [\ & \text{import } i; \\ & \text{export } e := e_0; \\ & \text{var } v := v_0; \\ & \text{do } A_1 [] A_2 [] ... [] A_n \text{ od} \\ ] \mid & \end{aligned}$$

The *import* section describes the imported variables i that are not declared, but used in A. The variables i are declared in other action systems, and thus they model the communication between action systems. The *export* section describes the exported variables e declared by A. They can be used within A and also within other action systems that import them. Initially, they get the values e_0. If the initialization is missing, arbitrary values from the type sets of e are assigned as initial values. The *var* section describes the local variables of action system A. They can be used only within A. Initially they are assigned values i_0, or, if the initialization is missing, some arbitrary values from their type set. Technically, all the used variables in import and export sections are global variables, and only variables defined in var section are local ones. The *do...od* section describes the computation involved in A. Within the loop, $A_1, \ldots, A_n$ are actions of A.

The behavior of the action system A is as follows: the execution starts by initialization of all variables, and then repeatedly, an enabled action from $A_1, \ldots, A_n$ is nondeterministically selected and executed. If two actions are independent, i.e., they do not have any variables in common, they can be executed in parallel [15]. Their parallel execution is then equivalent to executing the actions one after the other, in either order.

4.1.3. Composition of action systems

An action system is not usually regarded in isolation, but as a part of a more complex system. A large action system can be constructed from smaller ones using composition. Consider two action systems A and B below:

$$\begin{aligned} A = \mid [\ & \text{import } i; \\ & \text{export } e := e_0; \\ & \text{var } v := v_0; \\ & \text{do } A_1 [] A_2 [] ... [] A_n \text{ od} \\ ] \mid & \end{aligned}$$

$$\begin{aligned} B = \mid [\ & \text{import } j; \\ & \text{export } f := f_0; \\ & \text{var } w := w_0; \\ & \text{do } B_1 [] B_2 [] ... [] B_m \text{ od} \\ ] \mid & \end{aligned}$$

where $v \cap w = \phi$. We define the *parallel composition* of A and B, written A||B, to be the following action system C:

$$
\begin{aligned}
A \parallel B = |[\ & \text{import } k; \\
& \text{export } h := h_0; \\
& \text{var } u := u_0; \\
& \text{do } A_1[]A_2[]...[]A_n[]B_1[]B_2[]...[]B_m \\
& \text{od} \\
]|
\end{aligned}
$$

where $k = (i \cup j) \setminus h, h = e \cup f$ and $u = v \cup w$ .The initial values of the variables and the actions in A||B consist of the initial variables and actions of the original action systems.

The binary parallel composition operator || is associative and commutative and thus extends naturally to the parallel composition of a finite set of action systems. The behavior of a parallel composition of action systems is dependent on how the individual action systems interact with each other. The parallel composition operator can also be used in a reverse direction to decompose one action system into a number of those. More on these topics can be found elsewhere [15].

4.1.4. Refinement of action systems

A formal basis for the stepwise development of action systems is the *refinement calculus* [16]. In the refinement calculus, program statements are identified with their weakest precondition predicate transformers. However, the predicate transformer framework is not sufficient to reason about proactive systems. A *trace refinement* extension is described by Back and Wright [17] and *data refinement* extension by Sere and Waldén [18]. Our treatment of the action system refinement is based on the theory presented there.

4.2. Context models

With this formalism, we start modeling the context-aware and context-dependent systems by specifying the context-provider and context-utilizer roles as described in section 3. First we consider a context-dependent system, modeled by the action system CD:

$$
\begin{aligned}
CD = |[\ & \text{import } ... \\
& \text{export } ... \\
& \text{var } ... \\
& \text{do } g \to S[]\neg g \to T[]\beta \\
& \text{od} \\
]|
\end{aligned}
$$

Here g is the context guard and S is a statement dependent on the context g: $g \to S$ models the system behavior with provided context, and $\neg g \to T$ models the system behavior without provided context; β stands for the other actions of CD. The context guard g is a predicate on the local and context variable(s) x. A subset of the *import* and the *export* variables constitutes the context variables. The value of g is maintained by some other action system, called context-provider CP. Consequently, the context variable x is an *import*ed variable to CD and an *export*ed variable in CP.

Hence, we need to introduce the context provider, maintaining g in **Figure 2.** The context provider can potentially be a context-utilizer, depending on the service. If it were not a context-provider, there would not be any layer requiring handling of the context and it being the final consumer of the information. Thus, the provider is an independent, but necessary part of the system.

The context provider is modeled by action system CP: where b is a predicate; and $b \to x := x' | x' \in \{g, \neg g\}$ nondeterministically updates the global context variable x. The nondeterministic update is later refined to realistic intelligent algorithms. Hence, it models the context provided to CD.

$$
\begin{aligned}
CP = |[\ & \text{import } ... \\
& \text{export } ... \\
& \text{var } ... \\
& \text{do } b \to x := x' | x' \in \{g, \neg g\} \\
& \quad \neg b \to \text{V} \\
& \text{od} \\
]|
\end{aligned}
$$

Now, the parallel composition of action systems CD and CP, i.e. CD||CP is a complete context-aware model, and it models interactions between the context-provider and context-utilizer.

The implication of this model in the software architecture design can be explained in **Figure 2**, where the gray-shaded areas illustrate the main responsibility for the nodes belonging to them. The dark grey area constitutes the sensing nodes, the grey the en route nodes and the light-grey the gateway node. Moreover, it should be interpreted so that each item is considered belonging primarily to *layer* and secondarily to *segment*.

One merit of our model is that we intentionally separate the origin of the context from the whole context-aware system. This separation has one important consequence: the context is the result *after*

processing within the context-provider; i.e. the action system CP differentiates between *data* and *relevant data* and context is therefore always *refined* raw data.

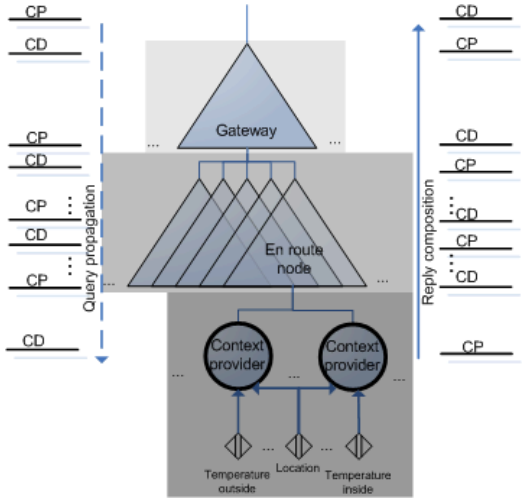

Figure 2. Data propagation / composition

As the realistic implication, the above idea contributes to a further classification of sensor nodes in wireless sensor networks as **Figure 2**. In this service oriented view, all sensors do not necessarily provide data needed for replying a query, nor does all function as en route nodes. Consequently, if possible the en route nodes decide based on the context whether their underlying sensing nodes can provide relevant information and thereby, forward them or not. The en route nodes can also, if implemented, compose data for providing relevance and energy efficiency. In the end, the context information is fused in the gateway node from the en route nodes to provide relevant and accurate answers for the propagated query.

4.3. Context refinement

In this section we discuss how the refinement principles can be used together with a parallel composition rule in our model. We show how to refine an abstract specification towards a detailed one, as well as the realistic implications of these refinements in system software design.

4.3.1. The context-utilizer

First, we consider one simple refinement scenario:

$$CD \| CP \leq_R CD' \| CP$$

where CD' is the refinement result of CD. The realistic implication of this scenario is upgrading the sensor application without touching the sensing part. This kind of refinement could mean: suppose we have a supervisory software CD running on top of the wireless sensor network infrastructure, now we update the existing software to a later version with more features CD'.

Since this category of refinement only concerns individual action systems, there should not be any change in the aggregated behavior of the whole system. Thus, we give the refinement rules as follows [17].

Consider two actions systems CD and CD':

$$\begin{aligned} CD = |[\ & \text{import } i; \\ & \text{export } e := e_0; \\ & \text{var } a := a_0; \\ & \text{do } A_1[]A_2[]...[]A_n \\ & \text{od} \\ ]| & \end{aligned}$$

$$\begin{aligned} CD' = |[\ & \text{import } i; \\ & \text{export } e := e_0; \\ & \text{var } a' := a_0'; \\ & \text{do } A_1'[]A_2'[]...[]A_n'[]X_1[]X_2[]...[]X_m \\ & \text{od} \\ ]| & \end{aligned}$$

where the local variables a in CD are replaced with new local variables a' in CD'. The actions A_i in CD are replaced with A_i' in CD', and auxiliary actions X_j are added into CD'.

R is a mapping relation between the new local variable a' and the old variable a. Consequently, we can say that the action system CD is refined by the action system CD', if there exists an abstraction relation $R(a,a')$ such that the following conditions hold:

1. Initialization: $R(a_0, a_0')$
2. Main actions: $A_i \leq_R A_i'$, for $i = 1,...,n$
3. Auxiliary actions: $\text{skip} \leq_R X_j$, for $j = 1,...,m$
4. Continuation condition: $R \wedge gCD \Rightarrow gCD'$
5. Internal convergence:
   $R \Rightarrow wp(\text{do } X_1[]X_2[]...[]X_m \text{ od}, \text{true})$

Here, the first condition says that the abstraction is established by the initializations. The second condition requires that each action A_i is refined by the corresponding action A_i' using $R(a,a')$. The third condition states that the auxiliary actions X_j behave like

skip with respect to the global variables $i \cup e$ while preserving *R(a,a')*. The fourth condition requires that an action in CD' is enabled whenever an action in CD is enabled and *R(a,a')* holds. The last condition stipulates that the execution of the auxiliary actions taken separately cannot continue forever whenever *R(a,a')* holds.

4.3.2. Refining the context variable

The other simple refinement scenario considers the context-provider itself:

$$CD \| CP \leq_R CD \| CP'$$

where CP' is the refinement result of CP. The realistic implication of this scenario is in improving the context processing unit without touching the upper layer sensor applications. This kind of refinement could be exemplified by for example: suppose we have a supervisory software running on top of the wireless sensor network infrastructure, now we improve the wireless sensor network infrastructure to provide more relevant and precise context information.

This category of refinement also concerns individual action systems and there is no change in the aggregated behavior of the whole system. Therefore, we can use the refinement rules described in section 4.3.1 in this case as well.

Here we consider one common refinement example on refining the context providing algorithm. In our initial model, the context providing algorithm is rudimentally expressed as $b \to x := x' | x' \in \{g, \neg g\}$. There is a need for further refining this algorithm into a realistic intelligent one. Usually this kind of refinement only refines *local actions*, more about this can be found elsewhere [18].

4.3.3. Compositional refinement

The last refinement scenario is a complex one, where the context-provider and context-utilizer *co-refine* together; i.e.,

$$CD \| CP \leq_R CD' \| CP'$$

where CD' is the refinement result of CD, and CP' is the refinement result of CP. The realistic implication of this scenario is refining the sensing part and application part simultaneously, interacting with each other. This kind of refinement could be exemplified as: suppose we have a supervisory software running on top of the wireless sensor network infrastructure, now we redesign the whole system, touching both the existing upper layer software and lower layer wireless sensor network infrastructure.

Obviously, this category of refinement is complex, because it concerns not only the individual behavior of each action system but also the aggregated behavior of the whole system [19].

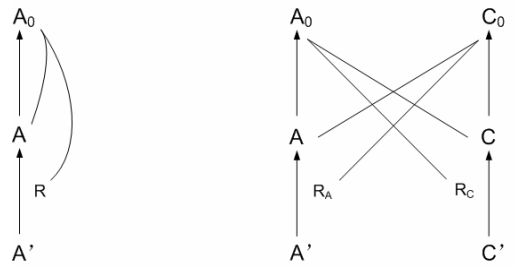

Figure 3. Individual refinement vs. compositional refinement

We can use the *compositional refinement* extension by Back and Wright [19], together with other refinement rules in section 4.3.1 and section 4.3.2, to refine this kind of scenario. In order to make the paper concise, we do not list down the complete refinement rules (more on these topics can be found [19]), but present an intuitive illustration for understanding this kind of refinement in **Figure 3**, where an arrow represents a refinement step and a line represents an abstraction relation.

Here we show an example of introducing new context to the whole system via compositional refinement: suppose we have the original system modeled as CD||CP, where CD and CP are defined in section 4.2. In this original setting, we have only *g* as our context. Now we would like to extend the context part by introducing a new context to the whole system. In reality, this scenario implies the case as utilizing additional data in the system which makes it necessary to redesign the system.

Using the compositional refinement, we can approach the problem as follows. First we consider the CD', which is the refinement result of CD. Let this new extra context be *d*. Assume *d* is a subset of $\neg g$, i.e. $d \subseteq \neg g$. Applying the refinement rules in section 4.3.1 and section 4.3.2, we can refine the original action $\neg g \to T$ in section 4.2 into two new actions

$$d \to R[](\neg g \setminus d) \to T'$$

where *R* and *T'* are refined statements satisfying

$$T \leq_R R \text{ and } T \leq_R T'$$

Then the new context V' is evaluated in CP', which is the refinement result of CP, declared in section 4.2.

$$\neg b \setminus d \rightarrow V'$$

Now CD'||CP' is the refinement result of CD||CP.

Actually this is an effective way of stepwise adding new features to the system, when simultaneously touching both the sensing part and the application part is inevitable. If we limit the *context* to *system failure*, this approach is similar to the work in [20] in which fault tolerance has been introduced to handle certain kind of faults.

5. Case study: from specification, via formalism, to implementation

We have implemented a smart kindergarten (nursery school) scenario as a case study for the proposed context-role categorization approach. The core concept of this application is illustrated in **Figure 4**, as a smart surveillance system for a kindergarten.

The system consists of stationary base stations, mobile sensor nodes which are attached to the children, and the supervisory application. The children are allowed to move freely in a predefined area (playground), and the supervisor is able to get the location information of all nodes (visually). When a child leaves the predefined area, the alertness level of the system increases, and the supervisor is informed. Higher alertness level implies intensified communication. Moreover, intensified location reporting, by the distinct node, is conducted when vibration is detected (the child is expected to be moving).

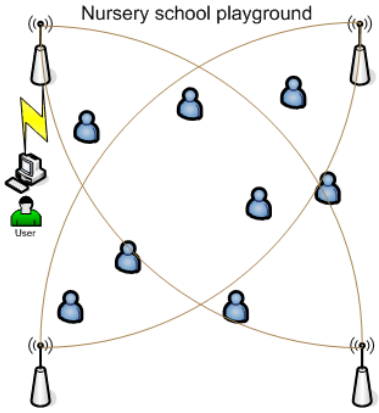

Figure 4. Smart kindergarten case study

This scenario is a typical context-aware and context-dependent example consisting of a context-provider and a context-utilizer. The system behavior, the context-utilizer, is critically dependent on different contexts provided by the context-provider, i.e. for supervision and localization. Moreover, in this particular example the base stations function as context-providers, the beacon, as well as context-utilizers, calculating the position and raising the alertness level.

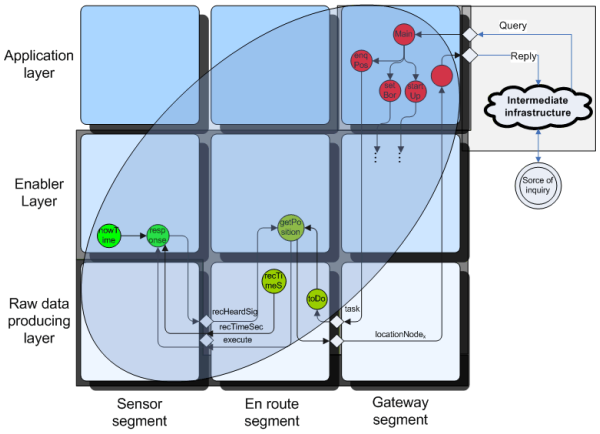

Figure 5. Final model of the system

Using the proposed context model in section 4, we implemented a variant of ROCRSSI [21] for the localizing service. Here we show a reduced model of the system in **Figure 5**, which is the stepwise developed result of **Figure 1**. This model works as the basis of the kindergarten application. The conclusion drawn was that the system is hierarchically pushing/pulling context information.

In order to make the paper concise, we elaborate a reduced system specification here, corresponding to **Figure 5**. A description of the kindergarten application and its implementation is available elsewhere [10].

The gateway segment on the application layer in **Figure 5** consists of a system *Main* which has been formalized as a composition of three subsystems *AenquirePosition*, *AsetBorders*, *AstartUp* and an interface system pushing data towards the inquirer. *Main* is below formalized as an action system. We take the subsystem *AenquirePosition* and its thread as our example.

The *Main* system is active on nodes belonging to the gateway segment. The subsystem *AstartUp* handles system the initialization, *AsetBorders* the definition process of the playground area and *AenquirePostion* request the position of a node (child). The system *AenquirePosition* fires when a user input of *locEnquiry* is detected. A variable called *task* is defined as a tuple space, functioning as the link to other action systems, defined underneath.

```
Gateway Segment
Main = AenquirePosition || AsetBorders || AstartUp

AenquirePosition = |[
userInput   locEnquiry;
export      task;
var         task = {(type,xa,yb,zc,flag)}…;
do
            task := task ∪ {(location, locEnquiry, y, z, false)})
od
]|
```

The En Route segment is active on the intermediate nodes functioning as "forwarders" in the system. It provides a service called *ToDo*. One instance in *ToDo* is *AgetPosition* that is context-dependent of the content in variable *task*. If *AenquirePosition* in the gateway segment is triggered, *AgetPostion* is also triggered. *AgetPostion* imports the *task* variable and in addition, a variable called *recHeardSig* that originates from the sensor segment. This action system also exports variables *execute* (imported to the sensor segment) and $locationNode_x$ (imported into gateway segment). Consequently, *AgetPostion* is CD upon *task* and *recHeardSig* but a CP for *execute* and $locationNode_x$. Here we do not give the specification of the En Route segment because of its complexity.

The specification for the sensor segment is shown underneath. The principles are the same as for the gateway segment.

```
Sensor Segment
Tracking= AsetCurrentState || Aresponse

Aresponse |[
import      execute, recTimeSec nowTime;
export      recHeardSig, execute;
var         recHeardSig∈ {(inqNodeID, (ids, dst, timeStamp))},
            stopTime ∈ Nat;
do
stopTime := nowTime + recTimeSec
if
nowTime < stopTime -> recHeardSig := recHeardSig
            ∪ (myID, (ids, dst, stopTime)) ∧
            (∀ execute.nodeID = myID : execute.flag = true)
fi
od
]|
```

The refinement has followed the ideas presented in this paper. The context variable is refined to be relevant for the user, providing an answer for the inquired task. For example, the amount of information for the user is limited if only a child's distance to its heard base stations would be provided. The relevance is increased by adding the location of the base stations and thus, the relative position of the child. This relation can be mapped into defined areas (*AsetBorders*).

Compositional refinement is conducted as soon as an imported / exported variable type is changed, that is when new functionality is added.

6. Concluding remarks

By taking a formal view of context-aware computing, we are able to reason about the foundational relationships that process context. A formal approach provides a framework for understanding the basic principles behind these various forms of interactions. In particular, our context model in this paper serves as a rigorous basis for the further development of a formal framework for design and evaluation of context-aware technologies.

Reference

[1] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. In *Proc. CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*, The Hague, The Netherlands, 2000.

[2] Mika Raento, Antti Oulasvirta, Renaud Petit, Hannu Toivonen. ContextPhone - A prototyping platform for context-aware mobile applications. In *IEEE Pervasive Computing*, 4 (2): 51-59, 2005.

[3] Special issue on Context-Aware Computing. *IEEE Pervasive Computing*, 2002.

[4] H. Chen, T. Finin, and A. Joshi. An ontology for contextaware pervasive computing environments. *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review*, 18(3):197.207, 2004.

[5] A. Schmidt, M. Beigl, and H.-W. Gellersen. There is more to context than location. Computers & Graphics, 23(6): 893-901, 1999.

[6] G. Chen and D. Kotz. A survey of context-aware mobile computing. Technical Report TR2000-381, Dartmouth College, Department of Computer Science, 2000.

[7] Lu Yan and Kaisa Sere. A Formalism for Context-Aware Mobile Computing. In *Proc. Third International Symposium on Parallel and Distributed Computing/Third International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Networks*, 2004.

[8] R.J.Back and K. Sere. From Action Systems to Modular Systems. In *Software - Concepts and Tools.* (1996) 17: 26-39.

[9] Mats Neovius and Christoffer Beck. From requirements via context-aware formalisation to implementation. In *Proc.*

the 17th Nordic Workshop on Programming Theory, Copenhagen, Denmark, 2005.

[10] Christoffer Beck. An application and evaluation of Sensor Networks. Master thesis, Åbo Akademi, Finland, 2005.

[11] S. Sitharama Iyengar and Richard R. Brooks. Distributed Sensor Networks. Chapman & Hall/CRC, 2004.

[12] Eiko Yoneki and Jean Bacon. A survey of Wireless Sensor Network technologies: research trends and middleware's role. Technical Report UCAM-CL-TR-646, University of Cambridge.

[13] Mats Neovius and Lu Yan. A Design Framework for Wireless Sensor Networks. To appear in *Proc. of IFIP 1st International Conference on Ad-Hoc Networking, Santiago De Chile, Chile.* 2006

[14] E. W. Dijkstra. A Discipline of Programming. Prentice Hall, 1976.

[15] R.J. Back and K. Sere. Stepwise Refinement of Action Systems. In *Structured Programming*, 12(1): 17-30, 1991.

[16] Ralph-Johan Back, Joakim von Wright. Refinement Calculus: A Systematic Introduction. Graduate Texts in Computer Science, Springer-Verlag, 1998.

[17] Ralph-Johan Back, Joakim von Wright. Trace Refinement of Action Systems. In *Proc. 5th International Conference Concurrency Theory*, Lecture Notes in Computer Science 836, Springer, 1994.

[18] Kaisa Sere, Marina A. Waldén. Data Refinement and Remote Procedures. In *Proc. Third International Symposium on Theoretical Aspects of Computer Software*, Lecture Notes in Computer Science 1281, Springer 1997.

[19] R. J. Back and J. von Wright. Compositional action system refinement. In *Proc. BCS FACS Refinement Workshop, Vol. 70 of Electronic Notes in Theoretical Computer Science*, Elsevier 2002.

[20] K. Sere and E. Troubitsyna. Hazard Analysis in Formal Specification. In *Proc. of SAFECOMP'99,* Toulouse, France, September 1999. Lecture Notes in Computer Science 1710, Springer Verlag.

[21] Chong Liu, Kui Wu, and Tian He. Sensor localization with Ring Overlapping based on Comparison of Received Signal Strength Indicator. In *Proc. IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, Oct. 2004.

[22] Paul Dourish. Where The Action Is: The Foundations of Embodied Interaction. MIT Press, 2001.

[23] Gruia-Catalin Roman, Christine Julien, and Jamie Payton. A Formal Treatment of Context-Awareness. In *Proc. 7th International Conference Fundamental Approaches to Software Engineering (FASE)*, Lecture Notes in Computer Science 2984, Springer 2004.

[24] Karen Henricksen and Jadwiga Indulska. A Software Engineering Framework for Context-Aware Pervasive Computing. In *Proc. 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2004.

[25] K. Henricksen. A framework for context-aware pervasive computing applications. PhD thesis, University of Queensland, Sept. 2003.

[26] A. Hinze, P. Malik, and R. Malik. Interaction design for a mobile context-aware system using discrete event modelling. In *Proc. Twenty-nineth Australian Computer Science Conference (ACSC), Hobart, Australia*, 2006.

[27] Ping Guo and Reiko Heckel. Modeling and Simulation of Context-Aware Mobile Systems. In *Proc. 19th IEEE International Conference on Automated Software Engineering (ASE)*, 2004.

[28] Andreas Pappas, Stephen Hailes, and Raffaele Giaffreda. A design model for context-aware services based on primitive contexts. In *Proc. UbiComp 2004.*

[29] Yannis Roussos and Yannis Stavrakas. Towards a Context-Aware Relational Model. Technical Report TR-2005-1, National Technical University of Athens, 2005.

[30] Eleftheria Katsiri. Middleware support for context-awareness in distributed sensor-driven systems. Ph.D. Thesis, University of Cambridge, Feb. 2005.

[31] Marco Carbone, Mogens Nielsen, and Vladimiro Sassone. A Formal Model for Trust in Dynamic Networks. BRICS Report RS-03-4, 2003.

Paper IV

Formal Modular Modelling of Context-Awareness

Mats Neovius and Kaisa Sere

Originally published in Formal Methods for Components and Objects, Frank S. Boer, Marcello M. Bonsangue, and Eric Madelaine (Eds.). Lecture Notes In Computer Science, vol. 5751. Springer-Verlag, Berlin, Heidelberg, 2009 pp. 102-118.

Formal Modular Modelling of Context-Awareness

Mats Neovius[1,2] and Kaisa Sere[1]

[1] Åbo Akademi University, Joukahaisenkatu 3 – 5, 20520 Turku, Finland
[2] Turku Center for Computer Science, Joukahaisenkatu 3 – 5, 20520 Turku, Finland
{mats.neovius,kaisa.sere}@abo.fi

Abstract. Characterising for a context-aware software is its ability to adjust to the prevailing situation. Such software reacts and bases the context-aware decisions upon inputs describing its operating conditions, i.e. on context(s). In this paper, we will seek the roots of context(s) and reason on the methods for deducing information by processing contexts; that is, present a methodology to enhance the relevance from raw data to knowledge. Thus, this paper will point out the relationship between introducing, constructing, serving, gluing and utilising context. Moreover, we show how to in a structured manner construct a context-service that satisfies given requirements and supplement the context-aware utiliser. For the sake of reuse and scalability, we will separate an application's specification from context reasoning and consider them as systems in their own rights. The findings will be motivated on a general level, with an easily conceivable example and formalised with the action system formalism.

1 Introduction

With the electro-mechanical development and the miniaturisation of transistors, the once fictitious deployment scenarios of computerised gadgetry turn into reality. As the computing is being weaved into the very foundations of our society, the domain of applicability extends. The reliance and expectations placed on these computerised gadgets are also ever increasing. Among others, gadgets are expected to be aware of the surrounding conditions and adapt automatically to them as envisioned by Weiser in 1991 [1]; that is, be context-aware. Because this development is likely going to continue, the future will be about navigating the ubiquity of information, being able to select, rely on and process relevant information [2, 3] as well as to reason rigorously with these.

Context in all its aspects complements software. As software alone is algorithmic and bound to operate on mathematical rules; the source of context in all its forms is data relying on some reading that characterise the operating conditions, e.g. temperature, location or identity. However, the contexts are ambiguous due to inherent inaccuracies of the acquiring equipments but are from the system's point of view unambiguous as no more descriptive data is available. Hence, context breaks the algorithmic model down [4] but introduces the possibility to context-awareness. Moreover, the provided contexts must be universal as no obligations on its utiliser aka. context consumer [5] or widgets [6], can be placed at time of creation. On the other hand, even though the application's algorithmic calculations were verifiable correct, misinterpreting a context is similar to

F.S. de Boer et al. (Eds.): FMCO 2008, LNCS 5751, pp. 102–118, 2009.

misinterpreting the operating conditions. Since context typically constitutes a decisive artefact, such misinterpretation can potentially result in faulty behaviour. We will however not consider faulty, absent, timeout or ambiguity of contextual information, as sheer fault tolerance and dependability issues branches to a separate field of research [7, 8, 9].

In paper we argue that a context-aware system cannot be said to be verified unless the construction and integration process of the necessary contextual information is. The sole reason is that discarding the treatment of context is intolerable for the sake of rigour, constituting the motivation of this paper. The main contribution addresses this source of motivation; this paper provides a methodology that will challenge the context (system) engineer to formally specify how the contextual information is constructed and integrated to a context-aware system that is to operate in a continuously changing contextual surrounding. That is, this paper is not about how to use context(s) but on what the context(s) constitute of, what are demanded from them and specifying how they are treated for providing rigorously to the required context-aware functionality.

Our approach takes an abstract view on the continuously changing context in a system. The contexts are considered globally available and thus, modelling the functional behaviour with shared variables suites our purpose well. Hence, we will concentrate on assuring the correct treatment of the provided (deduced) context. We treat context in a modular fashion defining an interface for the utiliser with which to depend on the contextual information through the glue that acquires and prepares contexts. This modularity is fundamental for the sake of adaptability [3], and hence also for scalability and reusability. Consequently, the context can be considered to be provided by a standalone, independent, replaceable and interoperable service. We use the action system formalism [10, 11, 12] to formally specify treatment of context, where the required syntactical language constructs are discussed in greater detail in Section 3.

We build on our earlier work [13, 14, 15] providing a methodology for integrating, depending on and formally treating continuously changing context. The context is represented by modules in separation from its utiliser alike in Context UNITY [3] that relates to our work but having an agent-like view on context-awareness with policies on updating the common context. In process calculi, Braione and Picco [16] consider an approach where inhibiting channels with context enables different implementations satisfying the same basic requirement whilst Zimmer [17] formalises, among others, a remote procedure call. Other approaches we are aware of [18, 19, 20] consider how a specification can be constructed given a rigorously modelled continuously changing environment, yielding a specification on the certain environment that it models.

The outline of this paper is as follows: in Section 2 we provide our definition of context and an example that is used throughout the paper. Section 3 introduces the action system formalism used to formally reason about context. Section 4 ties the context model with the action system formalism presenting how context is utilised, discovered, processed and composed for increasing the informative value. Finally, Section 5 concludes this paper.

2 Concepts Used in This Paper

We start by providing a definition of context and its different appearances. In Section 2.2 we outline an example to support the intuition of the reader when gradually referred to along with the formal definitions to various aspects that are provided throughout this paper.

2.1 Definition of Context and Context Related Matters

Research on context and context-awareness stems from 1992 and Olivetti's Active Badge research [21]. Following this, context has been given many and varying definitions. Pascoe [22] consider context to be subjective and defined by the entity that perceives it. Pascoe's subjectivity however refers to the perception made on the given context, such as 'close to'. Schilt et al. [23] considers aspects of context as "where you are, who you are with and what resources are nearby". Chen and Kotz [24] defines context to be environmental states and settings that affect the application and Yang and Galis [25] add the virtual object to the definition. Hence, according to these definitions context describe the operating conditions that have an impact on the application. As we concur with all, but further add the dictionary interpretations [26, 27] and Dey's and Abowd's [28], we end up in defining context accordingly:

> **Definition 1, context:** *Context is any information that can be used to characterise the situation of entities. An entity is a person, place, object, virtual object or state that is considered relevant to the interaction between a user and an application, including the user and the application themselves.*

Thus, according to the definition, context is a piece of information describing the situation of/in an entity that impacts the output/computations. Such context is typically extracted from either the logical e.g. identity, member of workgroup, time; or from the physical surroundings e.g. temperature, luminosity [29]. We do however not consider context to be cold, high, close, pretty, late or any other perceived matter.

In this paper, we call the source of contextual data *elementary context*. An elementary context is always from the system's point of view, a still-shot of the matter as it was at a specific moment. We call the outcome of composing contexts together and/or processing elementary contexts for providing enhanced information *deduced context*; which covers roles and relations of entities [30]. Consequently, we use the word *context* on a general level, whether it being an elementary or deduced context. The contexts are only updated by the entity introducing them. Given this definition and its interpretations, we define an activity or a system to be *context-aware* whenever any of its functionalities are impacted by some context per definition [28]. In other words, nearly all software reacting on some input could be considered context-aware to some extent [6].

The instance providing for the context is called a *context-service*. Thereby, a context-service is typically a careful composition of elementary context(s) that is considered an entity in its own right. The consumer of a service, the application or an intermediate compositional entity, is called the *utiliser* of this context-service.

In order for a context-service to provide some deduced contextual information, the service's output needs to be published. As an elementary context as such can potentially constitute a context-service in its own right, all context need to be published. Because all contexts are published, one context can provide to several context-services. For example, temperature at location x can be inquired by an utiliser, where translated to a Boolean (<20°C) as well as read to be used in some other service for calculating average temperature.

2.2 The Example: A Fictitious Speed Surveillance System

In order to motivate our ideas, we will construct a fraction of a simplified fictitious context-service providing the necessities for a speed surveillance system. The speed surveillance system is able to decide whether to allow further acceleration, qualifying as a good example encompassing straight forward decision making. The example demonstrate that once the algorithmic functionality of a context utiliser is verified, the hazards relate to the informal acquiring and perception of the information provided by context [31, 32]. It relies on easily conceivable calculations and on three distinct elements of contexts; namely one counting for current speed, one for the speed limit and one for whether the gas pedal position indicates acceleration. As the speed inevitably involves the logical context of time, we will show how to construct and integrate the context-service providing the perceived state of speeding, depicted in Figure 1.

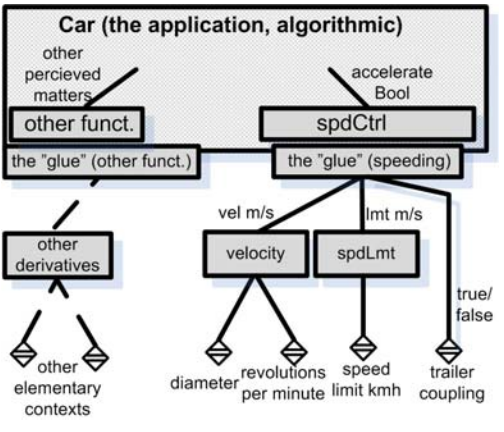

Fig. 1. Speed surveillance context architecture

In Figure 1, the bottommost "diamonds" depicts elementary contexts. The boxes compose and/or process the elementary context. Because the surveillance system is context-aware functioning in a continuously changing conditions where non-algorithmic events occur, exceptions to the functionality are implementable, depicted with the dashed lines and 'other' boxes. We show the adaptability of our approach by introducing the factor of a trailer coupling fixing the maximum speed limit. Examples basing on this surveillance system are clearly distinguishable in the text.

3 The Action System Formalism at a Glimpse

Formal methods facilitate systematic construction of reliable and rigorous software. Even though elementary contexts, as defined in this paper, are not software, formal treatment of them is important as they constitute in a decisive factor in the functionality of the context-aware software. Hence, not only the way contexts are integrated to software, but the methodology of composing deduced contexts from elementary context is of interest.

We model the construction and integration of contextual knowledge in the action system formalism. The action system framework provides means for reasoning about the contextual information in a modular, distributed, manner. For brevity, we omit type checking of the variables. Moreover, we aim at presenting a methodology rather than stepwise development, omitting the supported paradigm of refinement. Readers interested in the powerful methodology of refinement are directed to publications devoted to describing this [10, 11, 13, 33, 34, 35, 36, 37]. However, we feel obliged to stress that since refinement is about preserving correctness on mathematical foundations, it is restricted to the algorithmic part [4, 31, 32] and thereby, refinement as presented in the referenced literature, cannot be directly applied on the physical or logical elementary contexts.

3.1 Action System at a Glimpse

The action system framework is a state based formalism for defining distributed systems [12, 38]. It bases on Dijkstra's language of guarded commands [39, 40] and is defined with the *weakest precondition* predicate transformer, wp. From wp (A, q) we can derive all pre-conditions for which executing *action* A, the post-condition q is satisfied where pre and post-conditions are predicates over state variables. The weakest precondition is defined for various actions as follows:

wp (*abort*, q)	$= false$	*Aborting action*
wp (*magic*, q)	$= true$	*Miraculous action*
wp (*skip*, q)	$= q$	*Stuttering action*
wp ($x \coloneqq E$, q)	$= q[E/x]$	*Multiple assignment*
wp (A; B, q)	$=$ wp (A, wp (B, q))	*Sequential composition*
wp (A [] B, q)	$=$ wp (A, q) $\wedge$ wp (B, q)	*Nondeterministic choice*
wp ([a], q)	$= a \Rightarrow q$	*Assumption*
wp ({a}, q)	$= a \wedge q$	*Assertion*

The action *abort* is used to model disallowed behaviour, thus q is never satisfied, i.e. the outcome is *false*. Action *magic* always establishes true. Stuttering action *skip* does nothing, thus, the weakest pre-condition for establishing post-condition q is q. Action $x \coloneqq E$ is multiple assignment where every occurrence of x is substituted with an element from E. A; B is the sequential composition of two actions and A [] B the nondeterministic choice between actions A and B. [a] is the assumption and {a} is called the assertion. Assumption [a] is assumed true and {a} is a predicate needed to evaluate true in order for the execution to proceed to guarantee q. If assumption 'a' is *false*, the action behaves magically whilst if assertion 'a' evaluates false, the action aborts.

The language allows guarded commands, [g]; A, for convenience written $g \rightarrow A$, where g is the guard, the predicate and A the action, meaning in the wp-notation:

$$\text{wp}\ (g \rightarrow A, q) = g \Rightarrow \text{wp}\ (A, q)$$

that given the guard g, executing A satisfies q. The guard of A, gA is defined so that it does assure the establishment of a valid post-condition.

$$gA = \neg \text{wp}\ (A, false)$$

Having defined the guarded actions, we can define conditional choice and repetitive construct:

wp (**if** *A* **fi**, *q*) = wp (*A*, *q*) ∧ *gA*
wp (**do** *A* **od**, *q*)= (∀n.wp (A^n, *gA* ∨ *q*)) ∧ (∃n.¬*g*A^n)

where A^0 = *skip* and A^{n+1} = A^n; *A*. The repetitive construct defines that each action enables another or establishes *q* and that there must exist some that does not enable any other, i.e. partial correctness and termination. Within the repetitive construct, we define an action to only execute whenever its guarding predicate evaluates true.

To start reasoning with action systems, we define the elements of one, here named $\mathcal{A}$:

Definition 2, action system:
$\mathcal{A}$ = ![**var** *v*,*w**; **proc** *P*:*p*; *R**:*r*● *Init*: A_0; **do** *Lbl*:*A* **od**]!:*i*

In $\mathcal{A}$, *v* and *w** are the variables declared by this action system. Variables *v* are *local* and *w** constitute the uniquely named *export*ed variables (denoted with an asterisk). The clause **proc** defines procedures where *P*: *p* is a local procedure *p* labelled *P*, only executed if called upon whilst *R** is a uniquely named globally referable procedure. A procedure is substituted for each call on it from an action. Action *Init*:A_0 is the initialising action assigning the variables their initial value where *Init* is the label of this action, A_0. Each action and procedure label belongs to the *Names of labels* in the declaring action system. The **do**…**od** bracket pair constitutes the repetitive construct within which the action *A*, labelled *Lbl*, is repeatedly executed until *A* aborts or until termination i.e. when *gA* evaluates false; otherwise it continues infinitely. Whenever *gA* evaluates true, we say that the action is *enabled*. Of the enabled action(s) within the **do**…**od** clause, one is chosen non-deterministically for atomic execution. Variables *i* stand for the optional *import*ed variables that are declared and exported by other action systems but referenced from this. Together, import *i* and export *w** variables constitute a situation resembling shared writable memory between action systems.

This paper considers reactive action systems in which action system $\mathcal{A}$ is a part of a greater system where all other action systems are considered in their own rights but as $\mathcal{A}$'s environment, commonly denoted as $\mathcal{E}$ for environment. As the action atomicity holds on the greater system, an action of $\mathcal{A}$ can be preceded by an action in $\mathcal{E}$ impacting $\mathcal{A}$ by writing to $\mathcal{A}$'s global variable space. Consequently, in a reactive system a component does not terminate by itself as the environment can, through the global variables, enable some actions within this. This makes termination a global property and the formalism comes to showing properties of execution traces.

Distinct action systems can be composed according to Definition 3:

Definition 3, parallel composition '‖'**:** Let $\mathcal{A}$ and $\mathcal{B}$ be two action systems
$\mathcal{A}$ = ![**var** v_a, w_a*; **proc** *P*:*p* ● *Init*:A_0; **do** *LblA*: *A* **od**]! : *i*
$\mathcal{B}$ = ![**var** v_b, w_b*; **proc** *R**:*r* ● *Init*:B_0; **do** *LblB*: *B* **od**]! : *j*
Then, their compositional action system $\mathcal{C} = \mathcal{A} \parallel \mathcal{B}$ is
$\mathcal{C}$ = ![**var** v_m, w_n*; **proc** *P*:*p*; *R**:*r* ●*Init*:A_0; B_0 **do** *LblA*: *A*
[] *LblB*: *B* **od**]!: *h*
Where $h = i \cup j \setminus (w_a \cup w_b)$, $w_c^* = w_a \cup w_b$ and $v_c = v_a \cup v_b$ given that $v_a \cap v_b = \emptyset$.

In Definition 3, action system $\mathcal{C}$ is a parallel composition of $\mathcal{A}$ and $\mathcal{B}$. The definition basically states that if a set of action systems operate on disjoint set of local variables, $v_a \cap v_b = \varnothing$, procedure names and action labels, they can be composed to one action system where the actions within the repetitive **do** … **od** loop are treated non-deterministically and procedures remain intact. If the local variables are not disjoint or the local procedure names coincide, non-interference can be achieved through renaming. This compositionality provides a powerful means to formally compose and decompose action systems for abstraction and refactoring. In total, the action system framework provides us with a well established mathematically verified 'toolbox' with a sound semantic foundation to formally master modularisation, parallel composition, parallel and sequential execution, conditional and repetitive constructs.

3.2 Action Systems for Modelling Context

When modelling context, the import and export clauses do not suffice for passing of context due to the possibility of overwriting. Consequently, we introduce two new variable types for declarations of locally writable and globally readable variables: *read_only* and *publish* respectively denoted by a suffixing ◇, called *sentient* and *impact* variables by Roman et. al. [3]. Hence, advertising and reading the non-writeable context is possible, addressed in Property 1.

Property 1, context passing: Each *read_only* variable has exactly one system publishing it.

In addition, the introduction of elementary contexts motivates declaration of a special clause to the action system called **elemContext**, revising Definition 2 to 2'.

Definition 2', contextual action system:
$\mathcal{A}$=|[**elemContext** c; **var** v,w*, x◇; **proc** P:p; R*:r● $Init$:A_0; **do** Lbl:A**od**]|:i, y◇

In Definition 2', **elemContext** denotes the non-writeable elementary context c introduced by this action system whilst variables x◇ and y◇ denote the published and *read_only* variables respectively.

One elementary context can contribute to many deduced context. Thus, the action system introducing an elementary context needs to publish it as such, without alternation or processing, addressed in Property 2.

Property 2, introduction of context: Each elementary context is published as such.

The new variable types compel to revision of Definition 3 to 3':

Definition 3', parallel composition of contextual action systems '||':
Definition 3 with *read_only* variables v_a◇, v_b◇, v_c◇; *publish* variables w_a◇, w_b◇, w_c◇ and elemContext c_a, c_b and c_c for $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ respectively. Then:
v_c◇ = v_a◇ ∪ v_b◇\ (w_a◇ ∪ w_b◇), w_c◇ = w_a◇ ∪ w_b◇, c_c◇ = c_a◇ ∪ c_b◇
provided that ∀c_a∈ w_a and ∀c_b ∈ w_b.

Given these definitions and properties, we can denote contextual action systems and encapsulate its algorithmic calculations for verification. We exemplify this in example 1, omitting several pitfalls such as assurance of type checking.

Example 1: Consider three action systems, $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ calculating velocity based on revolutions in degrees per second (*rps*) and diameter.

$\mathcal{F}$ = |[**var** *vel*◊ ● *Init*:F_0;
do *Km/h*: *true* → *vel*◊ ≔ *rpm*◊ × *dia*◊ × π × 60 ÷ 1000 **od**]| : i_F, *rpm*◊, *dia*◊

$\mathcal{G}$ = |[**elemContext** *rps*; **var** *rpm*◊, *v*◊ ● *Init*:G_0;
do *RevPerMin*: *true* → *v*◊ ≔ *rps*; *rpm*◊ ≔ (*rps* ÷ 360 × 60) mod 1 **od**]|: i_G

$\mathcal{H}$ = |[**elemContext** *diameter*; **var** *dia*◊● *Init*:H_0;
do *WheelDia*: *true* → *dia*◊ ≔ *diameter* **od**]| : i_H

The action system $\mathcal{F}$ provides a service constituting of the deduced context velocity in km/h through the *publish* variable *vel*◊. *vel*◊ is calculated in the action labelled *Km/h*, given that the *read_only* variables are provided. Service $\mathcal{H}$ provides the diameter in meters and publishes this as *dia*◊ and $\mathcal{G}$ provides the service *rpm*◊ in revolutions per minute. Here, $\mathcal{H}$ merely maps the elementary context whilst $\mathcal{G}$ processes the elementary context *rps* to *rpm*◊. Hence, $\mathcal{G}$ and $\mathcal{F}$ function as the algorithmic part that is subjects to verification. Moreover, $\mathcal{G}$ publishes the elementary context *rps* unchanged as *v*◊. Unit concurrence, absolute vs. relative velocity, tolerance to mention a few are omitted. – *end of example*

In addition to the two types of variables and **elemContext**, we need to define means for the context utiliser to acquire this with unidirectional dependency, the glue. Thereby, we define a language construct called *dependency operator*, \\:

Definition 4, \\ dependency operator: Let A and B be two actions. Then, $A\backslash\backslash B$ is defined as: $A\backslash\backslash B = gA \wedge gB \rightarrow A; B$.

Definition 4 states the definition for \\ language construct denoting a dependency relation between two actions. This dependency relation is unidirectional, where both actions A and B need to be enabled and A guaranteed not to disable B[1] for $A\backslash\backslash B$ to be enabled. Mathematically, action B evaluates its guard gB prior to execution.

We will model the dependency on action/procedure labels in order to avoid confusion of concepts, i.e. $A\backslash\backslash B_{orig}$ in action system $\mathcal{D}$ where B_{orig} is the label of an action.

$\mathcal{D}$ = |[**var** *w*; **proc** *P*; ● *Init*: D_0; **do** *LblAdependB*: $A\backslash\backslash B_{orig}$ [] B_{orig}: *B* **od**]| : *i*

Declaring dependency between A and B directly restricts the expressiveness of action B to the inclusion of its guard as we cannot differentiate when action B is executed as a dependency reference and when as an action in its own right. Expressiveness is achieved by referencing a procedure instead of action B's label directly i.e. the action labelled *LblAdependB*: $A\backslash\backslash B_{orig}$ translates to $A\backslash\backslash P$ where P stands for a procedure that enables a specific variant of action B where the procedure action is substituted for the call on it. We label this specific variant B_{wake}. B_{wake} is executed once in the wake of a dependency reference, disables itself with a guard complementing gB_{orig}. Hence, the action labelled B_{orig} split up to two actions, B_{nat} and B_{wake}, making an action specifically

[1] The guard for $A\backslash\backslash B$: $\neg\text{wp}(A\backslash\backslash B, \mathit{false}) = gA \wedge gB \wedge \neg\text{wp}(A, \neg gB)$.

for dependency reference purposes. However, doing so breaks the atomicity of \\ and assurance of no other action disabling B_{wake} needs to be guaranteed, formally defined as atomicity refinement [10, 11].

$\mathcal{X}$ = |[… **do** *LblAdependB*: $A \backslash\backslash B_{orig}$ [] B_{orig}: B **od …**]|
-- *translates to* **–**
$\mathcal{X}$ = |[… **proc** P: $gB \wedge coord = false \rightarrow coord := true$
do *LblAdependB*: $gA \wedge gP \rightarrow A$; P
[] B_{wake}: $gB \wedge coord = true \rightarrow B$; $coord := false$
[] B_{nat}: $gB \wedge coord = false \rightarrow B$
od …]|

In the operational outline above, notable is that both B_{wake} and B_{nat} assure execution of action B, i.e. B_{orig} and the add-on guards exclude each other. The referenced procedure P's guard must include gB. The Boolean coordination variable *coord* assures that no dependencies are "pending"[2]. Procedure call substitution makes action labelled *LblAdependB* to execute the following:

LblAdependB: $gA \wedge gB \wedge coord = false \rightarrow A$; $coord := true$

For assurance of the transformation validity, the translation compliance with refinement ought to be shown. Indeed, the refinement calculus provides the conditions for auxiliary functionality to be added to B_{wake} and/or B_{nat}. Consequently, we have reached the situation of Definition 4 where given action $A \backslash\backslash B_{orig}$, A depends on an action labelled B_{wake} through the variables assigned by procedure P that guarantees execution of action B exactly once in the wake of action A.

In addition to \\, we define the @ operator to enable remote references in Definition 5.

> **Definition 5, @ construct location:** Let K label an action or a globally referable procedure and $\mathcal{K}$ an action system where $K \in$ labels of $\mathcal{K}$. Then $K@\mathcal{K}$ refers to action or globally referable procedure labelled K in action system $\mathcal{K}$.

Combining Definitions 4 and 5, writing in action system $\mathcal{A}$: $A \backslash\backslash K@\mathcal{K}$ [3] makes action A depend on an action labelled K in action system $\mathcal{K}$, providing for, for example, some deduced context. Recalling breaking of atomicity above, referring to a remotely available procedure is as follows where gP^* is the outcome of $gB \wedge coord = false$ and P is $coord := true$:

$\mathcal{X}$ = |[… **do** *LblAdependB*: $A \backslash\backslash P^*@\mathcal{Z}$ **od …**]|
-- *translates to* **–**
$\mathcal{X}$ = |[… **do** *LblAdependB*: $gA \wedge gP^* \rightarrow A$; P **od** …]|
$\mathcal{Z}$ = |[… **proc** P^*: $gB \wedge coord = false \rightarrow coord := true$
[] B_{wake}: $gB \wedge coord = true \rightarrow B$; $coord := false$
[] B_{nat}: $gB \wedge coord = false \rightarrow B$
od …]|

[2] Other data structures are implementable as well, such as queues, rings and so forth.
[3] Writing $A \backslash\backslash K^*@\mathcal{K}$ refers to a remote procedure.

Definition 3' is applicable for composition. Hence, if $K@\mathcal{K}$ provides a context, we have managed to successfully encapsulate the behaviour and construction of this contextual information and its updates from $A@\mathcal{A}$, just as intended, still complying with Definition 4. In the rest of the paper, we focus on showing how this separation of concerns can be exploited in a sensible manner.

4 Context Modelled with Actions Systems as a Part of a Program

As all software operates algorithmically, reasoning mathematically about its functionality is feasible and software can be shown to satisfy its requirements given that these are provided formally. When a system is formally verified, it explicitly meets with the formal requirements. Consequently, on a theoretical level, formally verified software on formally expressed requirements does not fail; it merely complies with its requirements.

Following the definition of context used in this paper, context and changes in it cannot be modelled formally as we cannot model the behaviour of the elementary contexts. However, putting effort into reasoning with context is motivated, as from a user point of view the reason for failing software, let it be misinterpretation or erroneous algorithm, is irrelevant as the consequences remain.

The aim of treating context in the presented modelling methodology is to reveal the characteristics of context to the designer for specifying them rigorously and verifying the involved algorithmic calculations. Because of this, we start by describing how a context-service is integrated to an utiliser, followed by describing how the elementary contexts are introduced. In Section 4.3 we show how these are formally treated to provide context information and provide a complete view of the characteristics.

4.1 Integrating Contextual Information to an Application

Claiming to have verified a context-aware system inevitably includes verification of its context. As the utiliser's context-aware decisions are impacted by *read_only* variables, a context-service can be treated as a black (white) box. Thereby, a context-service can be independently substituted for another, given that it provides the same verified contextual information on the same *publish* variables. This modularisation of contextual information facilitates reuse and provides comprehensibility through abstraction.

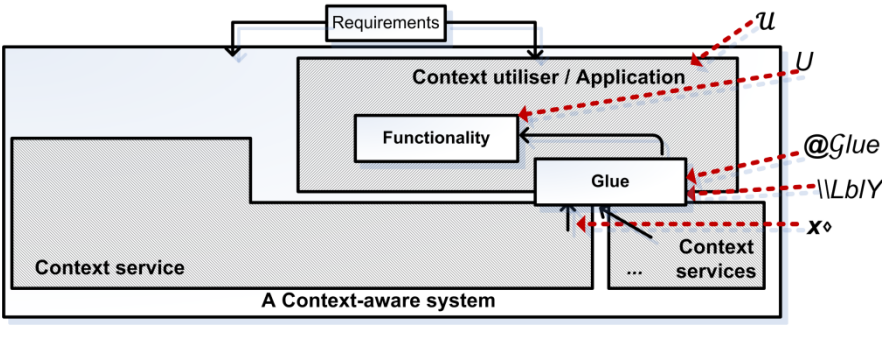

Fig. 2. Context-service - utiliser relation with references to example

A context-aware system can be depicted as in Figure 2, where the context utilising action system $\mathcal{U}$ depends on its glue to perceive matters based on certain context(s). Action U in $\mathcal{U}$ inquires an action or procedure in its glue, $LblY@\mathcal{G}lue$ to resolve some matter based on the *read_only* variable $x\diamond$ published by a context-service. The action initiating this, i.e. U, is only enabled given that the guard of $LblY@\mathcal{G}lue$ evaluate true. The dotted arrows and the labels in Figure 2 concur with action system $\mathcal{U}$ and $\mathcal{G}lue$ action U labelled $LblU$ and action Y labelled $LblY$ respectively, outlined below.

$\mathcal{U}$ =|[**var** w • $Init$:U_0; **do** $LblUdependonY$: U\\$LblY@\mathcal{G}lue$
[] '*other actions*' **od**]| : j

$\mathcal{G}lue$ =|[**var** y • $Init$:Y_0; **do** $LblY$: Y [] '*other actions*' **od**]| : i, $x\diamond$

Considering action system $\mathcal{U}$ to be the utiliser in Figure 2, it relies on action system $\mathcal{G}lue$ to glue. When so, guard gY is a predicate on the *read_only* variable(s) $x\diamond$ published by some other context-service. With this, we say that action system $\mathcal{G}lue$ perceives a feature of interest to $\mathcal{U}$.

The operators \\ and @ abstract the perception of context from the specific action that decides on it, i.e. $LblUdependonY$. This is essential as the utiliser cannot anticipate all operating conditions it will have to place decisions in throughout its lifetime [3]. Moreover, the *read_only* variable $x\diamond$ can be a prerequisite for several independent gluing action systems, facilitating scalability.

Example 2: Consider a speed surveillance system assembled in a car assuring that speeding will not take place, action system $\mathcal{U}$ in Figure 2. Because speeding is something that bases on speed limit and velocity, the system cannot proceed unless they are provided. Action system $\mathcal{G}lue$ counts for the glue, defining its action $LbLY$ as follows:

$\mathcal{G}lue$ =|[**var** ; • $Init$:Y_0;
do $LblY$: $vel\diamond \leq spdLmt\diamond \rightarrow Y$
[] '*other actions*'
od]| : i, $vel\diamond$, $spdLmt\diamond$

where $vel\diamond$ refers to velocity as calculated in example 1 and $spdLmt\diamond$ to speed limit that are updated and published by some context-service. According to Definition 4, U\\$LblY@\mathcal{G}lue$ is to be enabled if gU is true and speeding is false, resolved in the guard of action labelled $LblY$. As maximum velocity is fixed whenever a trailer is coupled, including the Boolean $trailCpl\diamond$ according to Figure 1, action labelled $LblY$ must treat this for the whole range of values. Consequently, action system $\mathcal{G}lue$ becomes:

$\mathcal{G}lue$ =|[**var** ; • $Init$:Y_0;
do $LblY$: $((vel\diamond \leq spdLmt\diamond \wedge \neg trailCpl\diamond) \vee$
$(vel\diamond \leq spdLmt\diamond \wedge trailCpl\diamond \wedge vel\diamond \leq 80\text{kmh})) \rightarrow Y$
[] '*other actions*'
od]| : i, $vel\diamond$, $spdLmt\diamond$, $trailCpl\diamond$

Notable is that the utiliser needs only to rely on that the action system $\mathcal{G}lue$ indeed provides adequate velocity. Having actions in the glue raising specific flags whenever certain condition are met abstracts the evaluation of sometimes long guards from the utiliser – *end of example*

Because the utiliser $\mathcal{U}$ and the glue $\mathcal{G}lue$ are treated independently from the context-service providing $x\diamond$, the service must not pose any obligations on how its reading is to be perceived. For the context $x\diamond$ we cannot allow confusion between a valid "*context value*" and the absence/timeout of it, i.e. "*do not know*". The absence/timeout refers to erroneous or outdated context that as noted earlier, is out of the scope of this paper.

We define context universality for valid values, Definition 6:

Definition 6, context universality: Let c_n denote the domain of a context and c_m the range decided on, where $c_m \subseteq c_n$ and let c_i be the complement of c_m. Then the context-service must provide for c_i as well.

Since the nature of context, the utiliser becomes a coordinating system that triggers some functionality based on current context(s). The impact of a context can be tuned with non-contextual information in the referencing action $\mathcal{U}$, for example, scheduling action U in action system $\mathcal{U}$ or prioritising it over another [41].

4.2 Composing Information from Elementary Contexts

The elementary contexts constitute the basis for all deduced contexts and context-awareness, making the process of constructing a context-service seemingly hierarchical. Figure 3 depicts any level in the process of constructing a context-service. The input data to this level, the context dependent (CD) segment aka. context provider [5], takes the elementary context c introduced here and/or some *read_only* variables $y\diamond$ as inputs, publishing it as $z\diamond$. $z\diamond$ is then processed in the context refiner/reasoner (CI) segment (aka. context synthesizer [5]). The output is published by the providing (CP) segment [13]. We define the segment interdependencies as follows, omitting type checking:

Definition 7, acquiring context CD: Let CD *read_only* $y\diamond$, introduce elementary context(s) $c \subseteq c_n$ and publish $z\diamond$ and $r\diamond$, then $z\diamond \subseteq y\diamond \cup c$ and $r\diamond = c$.

Definition 8, improving context CI: Let CI *read_only* $z\diamond$ and publish $q\diamond$, then $q\diamond := f(z\diamond)$ according to refiner/reasoner involving optional imported variable conditions i.

Definition 9, providing context CP: Let CP publish $x\diamond$ and *read_only* $q\diamond$, then assuming $q\diamond$ is published by the CI and $r\diamond$ is the set of elementary context(s) introduced by this processing level, $x\diamond := q\diamond \cup r\diamond$ and i be updated.

Hence, the output of this processing level is $x\diamond := f(z\diamond \cup c) \cup r\diamond$ given that the necessary input is provided. Writing this as action systems, the three segments in Figure 3 and Definition 7 through 9 translate into namesake action systems $\mathcal{CD}$, $\mathcal{CI}$ and $\mathcal{CP}$.

$\mathcal{CD}$ =|[**elemContext** c; **var** $z\diamond$, $r\diamond$● CD_0;
do *Get*: *true* → $z\diamond := y\diamond \cup c$, $r\diamond := c$ [] '*other actions*' **od**]| : $y\diamond$

$\mathcal{CI}$ =|[**var** $q\diamond$, β; ● CI_0; **proc**;
do *Process*: *true* ∧ i → $q\diamond := f_1(z\diamond, i)$[] '*other actions*' **od**]| : i, $z\diamond$

$\mathcal{CP}$ =|[**var** $x\diamond$; ● CP_0; **do** *Provide*: *true*→ $x\diamond := q\diamond \cup r\diamond$ [] '*other actions*' **od**]| : i, $q\diamond$, $r\diamond$

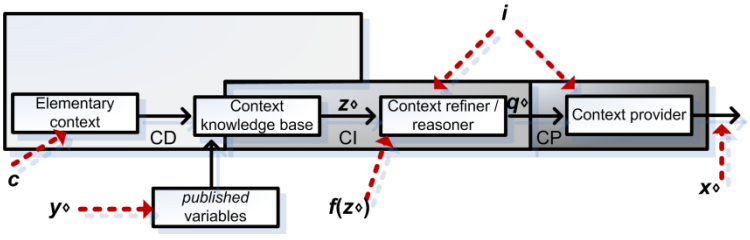

Fig. 3. Processing context

The action system labelled $\mathcal{CD}$ handles the introduction of the variables, the $\mathcal{CI}$ the actual algorithmic functionality and the $\mathcal{CP}$ the publishing of the deduced context(s) and the possible elementary context(s). The *import* variables *i* provide the possibility for shared variables, e.g. asynchronous handshaking.

This segmentation defines input and output interfaces and encapsulates the algorithmic part. At the same time, the elementary context(s) is available as measured to be included by other systems. Combined with the *read_only* variables, the processing increases the level of information that is eventually published.

4.3 Processing Context

The task of constructing a context-service providing the context read by the glue reveals the importance of mastering the composition and calculation with context. Recalling Figure 3, one instance of context processing, Figure 4 illustrates the relation of several such instances resulting in context services providing for action system $\mathcal{Glue}$ in Section 4.1.

Figure 4 depicts how the en route context improvers increase the relevance depending on *publish* variables [2, 13] and elementary contexts. Hence, guaranteeing loop freeness of the context variables is necessary; declaring that the *publish* variable(s)

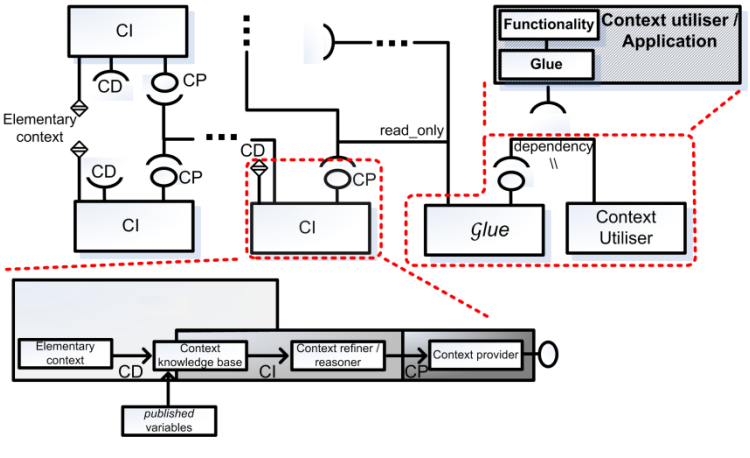

Fig. 4. Context processing

that are *read_only* to a certain level must not include that same level's published variables nor a deduced context depending on such constituting in Property 3.

Property 3, loop avoidance: Let an instance *read_only* $y_n\diamond$ relying on *publish* variables $t_n\diamond$ and $\alpha := y_n\diamond \cup t_n\diamond$. Then α denote all variables this instance relies on. Let c denote elementary contexts introduced by this instance and $x\diamond$ variables it publishes, then $\alpha \cap x\diamond = \emptyset$ and $x\diamond$ comes to rely on $\alpha \cup c$.

In addition to Property 3, in order to provide well defined abstractions and verifiable deduced context, keeping track of the context unit(s) is important.

With these restrictions, processing context is the act of increasing the relevance of information by applying some algorithm or composing several contexts together. Each context processing level, as there might be several (denoted by three dots in Figure 4), is alike the one depicted within the dotted lines down left in Figure 4 and in Figure 3. The context utiliser, in upper right corner Figure 4, is as the dependency references depicted in Figure 2.

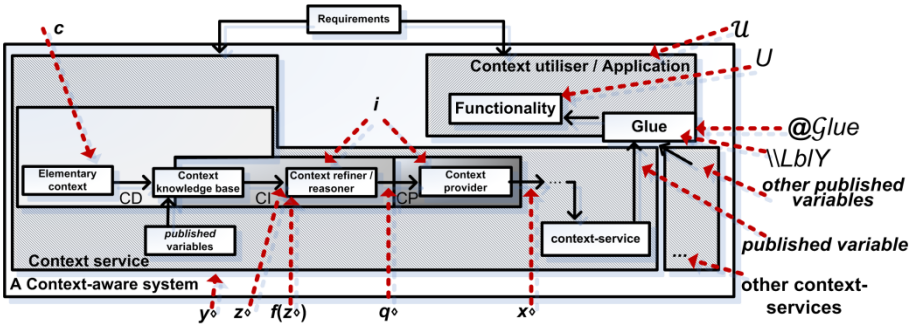

Fig. 5. Construction of a context-service

Example 3: Considering example 1 and 2 and Figure 4, the unit of velocity and speed limit must coincide. The three CI boxes in Figure 4 could stand for action systems $\mathcal{F}$, $\mathcal{G}$ and $\mathcal{H}$ in example 1. The utiliser's names correspond to names used in Section 4.1. Moreover, for the sake of reuse, the system must take a stand on the units and their implementation, such as whether the velocity is absolute or relative – *end of example*.

Figure 5 combines all presented the figures depicting the processing of context to a context-aware system. The Definitions 4 through 9 presented in this paper assure that contexts place no obligation on its utilisers and that it can be reasoned about like if it was a special variable with restricted write access.

5 Conclusions

This paper stresses the importance of processing contextual information systematically as context most certainly constitutes a decisive factor of any context-aware system. Because of this, in order to claim that a system is formally verified, we argue that

the decisive matters, including context and its processing, need to be formally expressed and its mathematical matters verified. In this paper, we have presented a methodology and a language construct to the action system formalism that split the contextual characteristics from the software through a gluing system. The contexts are considered to be provided and processed within context-services. We have also outlined and motivated qualities of a context variable that need to hold for facilitating scalability and reuse.

Modelling context in the presented methodology challenges the designer to construct rigorous realistic context-aware systems. This is achieved by revealing the characteristics of the needed context when formally specifying the processing of context utilised by an application. Once these contexts are formalised, the formalisation has fulfilled a purpose of revealing shortcomings to the designer. The action system framework is used for processing and composing contexts where the constraints are placed by the elementary context. Moreover, as this paper consider modularised context, we can foresee that the presented ideas could be extended to formalise other distributed well-defined matters as well.

Being able to express dependencies between actions and services is a first step in modelling services with action systems; future work will address chains of dependencies, unordered dependencies as well as showing characteristics of refinement of inter-dependent actions. We aim at instead of having a library of model transformation rules, to define new simple language construct with which expressing the challenges brought along with the ever increasing distribution of computations and responsibilities are possible.

Acknowledgements. Mr. Neovius wishes to express his gratitude towards TOP-säätiö for the financial support he has received. The authors' wishes to thank to Mr. Fredrik Degerlund for the discussions and comments and the FP7 IST-2007.1.2 DEPLOY-project for partly funding this research. Moreover, a special thank goes to the reviewers for extraordinary extensive and valuable comments on means to improve this paper.

References

1. Weiser, M.: The Computer for the Twenty-First Century. Scientific American (1991)
2. Neovius, M., Yan, L.: A Design Framework for Wireless Sensor Networks. In: Proceedings of the IFIP 19th World Computer Congress (2006)
3. Roman, G.-C., Julien, C., Payton, J.: A formal treatment of context-awareness. In: Wermelinger, M., Margaria-Steffen, T. (eds.) FASE 2004. LNCS, vol. 2984, pp. 12–36. Springer, Heidelberg (2004)
4. Shaw, M., Garlan, D.: Software Architecture, Perspectives on an Emerging Discipline. Prentice-Hall Inc., Englewood Cliffs (1996)
5. Ranganathan, A., Al-Muhtadi, J., Campbell, R.H.: Reasoning about Uncertain Contexts in Pervasive Computing Environments. IEEE Pervasive Computing 3(2) (2004)
6. Dey, A.K., Abowd, G.D., Salber, D.: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction 16(2) (2001)
7. Aviziens, A.: Fault-Tolerant Systems. IEEE Transactions on Computers C-25(12) (1976)

8. Randell, B., Lee, P., Treleaven, P.C.: Reliability Issues in Computing System Design. ACM Computer Survey 10(2) (1978)
9. Avizienis, A., Laprie, J.-C., Randell, B.: Dependability and its Threats: A Taxonomy. In: Proceedings of the IFIP 18th World Computer Congress (2004)
10. Sere, K., Waldén, M.A.: Data Refinement and Remote Procedures. In: Ito, T., Abadi, M. (eds.) TACS 1997. LNCS, vol. 1281. Springer, Heidelberg (1997)
11. Sere, K., Waldén, M.A.: Data Refinement of Remote Procedures. Formal Aspects of Computing 12(4) (2000)
12. Back, R.J.R., Kurki-Suonio, R.: Decentralization of Process Nets with Centralized Control. In: Proceedings of the 2nd ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (1983)
13. Neovius, M., Sere, K., Yan, L., Satpathy, M.: A Formal Model of Context-Awareness and Context-Dependency. In: Proceedings of the 4th IEEE International Conference on Software Engineering and Formal Methods (2006)
14. Degerlund, F., Sere, K.: A Framework for Incorporating Trust into Formal Systems Development. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) ICTAC 2007. LNCS, vol. 4711, pp. 154–168. Springer, Heidelberg (2007)
15. Yan, L., Sere, K.: A Formalism for Context-Aware Mobile Computing. In: Proceedings of the Third international Symposium on Parallel and Distributed Computing/Third international Workshop on Algorithms, Models and Tools For Parallel Computing on Heterogeneous Networks (2004)
16. Braione, P., Picco, G.P.: On Calculi for Context-Aware Coordination. In: De Nicola, R., Ferrari, G.-L., Meredith, G. (eds.) COORDINATION 2004. LNCS, vol. 2949. Springer, Heidelberg (2004)
17. Zimmer, P.: A Calculus for Context-Awareness. BRICS Report Series RS-05-27, Denmark (2005) ISSN 0909-0878
18. Petre, L., Qvist, M., Sere, K.: Distributed Object-Based Control Systems. Technical Report 241, TUCS (Feburary 1999)
19. Rönkkö, M., Ravn, A.P., Sere, K.: Hybrid Action Systems. Theoretical Computer Science 290(1) (2003)
20. Hayes, I.J., Jackson, M.A., Jones, C.B.: Determining the specification of a control system from that of its environment. In: Proceedings of the International Symposium of Formal Methods (2003)
21. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The Active Badge Location System. ACM Transactions on Information Systems 10 (1992)
22. Pascoe, J.: Adding Generic Contextual Capabilities to Wearable Computers. In: Proceedings of the Second International Symposium on Wearable Computers (1998)
23. Schilit, B., Adams, N., Want, R.: Context-Aware Computing Applications. In: Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (1994)
24. Chen, G., Kotz, D.: A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College (2000)
25. Yang, K., Galis, A.: Policy-Driven Mobile Agents for Context-Aware Service in Next Generation Networks. In: Horlait, E., Magedanz, T., Glitho, R.H. (eds.) MATA 2003. LNCS, vol. 2881, pp. 111–120. Springer, Heidelberg (2003)
26. Merriam Webster Online dictionary, Merriam-Webster Inc., Springfield, MA 01102
27. Oxfords Advanced learner's dictionary (2000) CD-ROM
28. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: Proceedings of the CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness (2000)

29. Schmidt, A., Aidoo, K.A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., Van de Velde, W.: Advanced interaction in context. In: Gellersen, H.-W. (ed.) HUC 1999. LNCS, vol. 1707, p. 89. Springer, Heidelberg (1999)
30. Coutaz, J., Crowley, J.L., Dobson, S., Garlan, D.: Context is key. Communications of the ACM special issue: The disappearing computer 48(3) (2005)
31. Zemanek, H.: Abstract Architecture, General concepts for systems design. In: Bjorner, D. (ed.) Abstract Software Specifications. LNCS, vol. 86. Springer, Heidelberg (1980)
32. Naur, P.: Intuition in software development. In: Ehrig, H., Floyd, C., Nivat, M., Thatcher, J. (eds.) TAPSOFT 1985 and CSE 1985. LNCS, vol. 186. Springer, Heidelberg (1985)
33. Back, R.J.R., Sere, K.: Stepwise Refinement of Action Systems. Structured Programming 12(1), 17–30 (1991)
34. Back, R.J.R., von Wright, J.: Refinement Calculus: A Systematic Introduction. Graduate Texts in Computer Science. Springer, Heidelberg (1998)
35. Back, R.J.R., von Wright, J.: Trace Refinement of Action Systems. In: Jonsson, B., Parrow, J. (eds.) CONCUR 1994. LNCS, vol. 836. Springer, Heidelberg (1994)
36. Back, R.J.R., von Wright, J.: Compositional Action System Refinement. TUCS technical report no. 464 (June 2002)
37. Back, R.J.R.: Correctness Preserving Program Refinements: Proof Theory and Applications. Mathematical Center Tracts, vol. 131, Mathematical Centre, Amsterdam, The Netherlands (1980)
38. Sere, K.: Stepwise derivation of parallel algorithms, PhD dissertation, Åbo Akademi (1990)
39. Dijkstra, E.W.: A Discipline of Programming. Prentice Hall, Englewood Cliffs (1976)
40. Dijkstra, E.W.: Guarded commands, nondeterminacy and formal derivation of programs. Communications of the ACM 18(8) (1975)
41. Sekerinski, E., Sere, K.: A Theory of Prioritizing Composition. The Computer Journal 39(8) (1996)

Paper V

Mastering the Relevance of Subjective Information in Ubiquitous Computing

Mats Neovius and Kaisa Sere

Submitted to International Journal of Networked Computing and Advanced Information Management (IJNCM) Special issue on Social Informatics and COMputing (SICOM).

Mastering the Relevance of Subjective Information in Ubiquitous Computing

[1, *2] Mats Neovius, [2] Kaisa Sere
[1] *Turku centre for Computer Science, Joukahaisenkatu 3-5, 20520 Turku Finland, E-mail: firstname.surname@abo.fi*
[2] *Åbo Akademi University, Department of Information Technology, Joukahaisenkatu 3-5, 20520 Turku, Finland, E-mail: firstname.surname@abo.fi*

Abstract

An application that relies on a ubiquitous computing environment populated by autonomous software agents is saturated by information availability. When this information is subjective, to master the (ir)relevance of it, this paper formally defines group that bond by likes. A group is a set of software agents. The likes and their bonding are captured by comparing the frequency and character of the experiences on some provider. Because the group bond by likes, the experiences of a group an inquirer associates itself with are considered relevant information. These experiences are calculated with and means to compose, customise and define abstract groups are provided. For this, the Subjective Logic framework capturing a degree of certainty in addition to (ir)relevance is chosen. Hence, this paper proposes a methodology for abstracting sets of software agents to groups that capture the subjective experiences of a proposition by likes. This constitutes the key to master relevance of information.

Keywords: *reputation based trustworthiness, subjective logic, collaboration, information relevance*

1. Introduction

Ubiquitous computing is called the third wave of computing, a successor to distributed computing and mobile computing [1]. It is characterised by technologies that weave themselves indistinguishably to the everyday life [2]. Moreover, we consider it in this paper distributed, open, of a structure that is ever changing and being populated by interacting autonomous software agents. These software agents produce and process information to be consumed by an application that provides the user a means to perform a task [3]. The consumed information is considered subjective because of the autonomy of the providing software agents and the inherent inaccuracy of the source of information, i.e. not knowing the intents of the providing subject(s), their frequency of error or the context of the view. The level of subjectivity on the acquired information is captured by user specific experiences. A user specific experience is a cognitive evaluation by the user on the providing subject in a proposition, e.g. *A*'s experience in subject *B* in proposition *serves tasty food*. These user specific experiences are recorded and constitute a user's history. The history forms a user's profile that is considered a software agent in its own right. Hence in accordance to [4, 5, 6, 7, 8] and depicted in **Figure 1**, we separate concerns between information providing and functional segments.

Considering a user's history of experiences at a given moment and the ever changing nature of the ubiquitous computing environment, a user's experiences may be used to estimate the degree of truth on the subject software agent in a proposition. This degree of truth involves a level of (un)certainty due to the changing ubiquitous computing environment. To mend the uncertainty, a user may ascertain its degree of truth by referring to other users' experiences. However, due to the subjectivity of the experiences, only users' experiences who share the likes are considered relevant. The bonding by likes is determined by the similarity of the users' profiles where the bonded software agents form a social group on a subject in a proposition.

The outline of the paper is as follows. In Section 2 a general model to record the experiences of subjects in a proposition is presented. This is fundamental to express experiences in a subject providing for multiple propositions, e.g. *B* may provide for *serves tasty food* and *relaxing atmosphere*. Section 2 also defines the type of an experience used in this paper. The presented model of an experience and their histories may be mapped to the type required by the Subjective Logic (SL) framework. Section 3 presents functions of the SL framework [9, 10, 11, 12]

including the mapping function in Section 3.2. Section 4 proposes the contributions of this paper. Firstly, it proposes a formal definition of a group. A group is considered a set of software agents bonded by likes on a subject in proposition. It is considered a software agent in its own right. Hence, a group provides experiences and a hierarchy of groups is expressible. The means to derive these experiences are examined. The section also outlines how such groups may be composed with respect to a Boolean operation, defining a customised group. Such grouping of software agents is, to the best of our knowledge, novel. Secondly, as each experience is of equal weight, we propose a view of distributing the experiences on software agents involved in acquiring the understanding. After these contributions, Section 5 provides an example followed by a discussion in Section 6 and conclusions in Section 7.

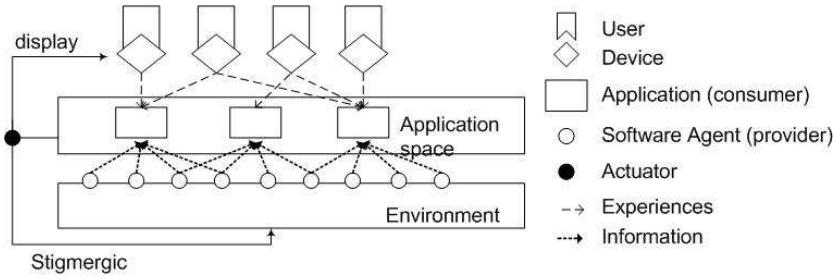

Figure 1. Setting of this paper

As a consequence, this paper considers the (ir)relevance of a subjective piece of information to be a synonym to experience based trust evaluating its (un)trustworthiness. That is, a provider provides a subjective piece of information whose relevance to the user is defined by the user's experiences. Both (ir)relevance and (un)trustworthiness aim at resolving a level of reliance on a subject in a proposition by observing the history. Moreover, as initially there is no experiences, (un)trustworthiness / (ir)relevance is something that builds up from initial ignorance (do not know). This underlines the importance of expressing the level of ignorance, hereafter called (un)certainty. This gives rise to a three-valued parameter further motivating the choice of the SL framework. We consider information derivation and relations between propositions (ontology) out of the scope of this paper. Moreover, we will hereafter use consistently the term (un)trustworthiness over (ir)relevance.

2. Trustworthiness and its formation

The ability to trust is a cornerstone for the existence of the human society [13]. In cognitive sciences, it is a mental state that enables collaboration, formation of groups, feeling of relative security etc. [14]. Moreover, trust enables a feeling of reliance in a matter, e.g. trust on the babysitter to take care of the children. This 'feeling' is something that only cognitive agents having internal explicit goals and beliefs may perceive in some other uniquely identifiable matter [14, 15]. In computer science, trust has been realised as policy and reputation based trust [16]. The latter is sometimes considered a subset of the former, as ultimately the decision is Boolean, i.e. a policy that weigh between risk and profit. Hence, experience based trust is a means to provide the decision maker with data to make a better, more satisfactory decision. In addition, a third form called social trust has been presented [17], but we consider this an instance of reputation based trust.

Policy based trust, also called resource access trust [18], was originally introduced by Blaze et al. [19] and relies on logical rules to enforce trust, typically realised as a predicate. Implementations of policy based trust includes access control, firewall rules, authorizations etc. In the policy based setting, inaccurate or incomplete information are not allowed or considered rudimentary, i.e. as complete and correct. Hence, policy based trust is suitable in environments where assuming complete and correct behaviour is motivated and is sometimes considered as a branch of security. As of this, policy based trust could be weaved into a formal model. Because this paper has a focus on the trustworthiness of subjective information, policy based trust will not be considered any further in this paper.

Reputation based trust, on the other hand, is similar to the human notion of trust. It is used interchangeably with the term experience based trust, which we prefer hereafter. The kind of

experience based trust considered in this paper is dynamic, incomplete, subjective and it builds up. Therefore, in addition to trust and distrust, the initial level of uncertainty for 'do not know' is to be captured, requiring a three-valued parameter. Following [20], also we consider confusion between 'do not know' and 'do not trust / trust' intolerable. Moreover, complete certainty may never exist as of the changing environment. Hence, the goal with experience based trust is to provide the subjective probability with which an agent assesses that the target will perform according to expectations [21] by examining past behaviour. Implementations of experience based trust include, but are not limited to, online auctions (eBay.com), product review sites (Epinions.com) and discussion forums (SlashDot karma) [22]. Hence, for the application to evaluate a level of trust on a subject in a proposition, the trust is necessarily experience based. Hence, experience based trust is merely a means to provide the decision maker with data to make a better, more satisfactory decision.

We define trust according to McCarthy and Chervaney [23]:

> Definition of Trust: "*The extent to which one party is willing to depend on the other matter in a given situation with a feeling of relative security, even though negative consequences are possible*".

This definition implies that expressing trust in something certain is void, making trust viable whenever evaluating something not Boolean, i.e. uncertain or dogmatic. Moreover, the definition considers dependence, reliability and motivates considering risk as a factor in the decision [15]. The subsequent sections elaborate on the properties and relations between the trustor and the trustee, motivate the foundation of the (un)certainty, provide a representation of it as well as outline the restrictions on a network of trust.

2.1. Conditions and Logical Properties of Trust Relations

To establish a trust relation, unique identification of the counterpart is necessary. Assuming this and complete trust in oneself, trust, distrust and uncertainty together denote a level of reliance the source (trustor) places on the trustee. This level of reliance is subjective and may be ascertained by inquiring referrals. That is, consider software agents A, B and C, then trustor A may inquire trustee B for its experiences on subject C in a proposition x to ascertain its level of reliance.

This is called trust transitivity and exemplifies the need of discounting reliance, i.e. A discounts B's view on C in x by A's view on B in recommending to x. Trust transitivity is argued against by Christianson et al. [24]. However, we claim that trust transitivity is feasible with certain restrictions. The restrictions stem from the definition of trust where expressing distrust as opposed to trust is possible. Thereby, transitivity by distrust (*disbelief*) is argued as of the binary relation of transitivity [25], e.g. if A distrusts B and B distrust C in say x; then, does this imply that A should trust C in x? Hence, we restrict transitivity to 'positive' trust. The positive trust (*belief*) delegation of A trusting B to recommend where B recommends C in x is captured; where A provides B with the (partial) power to decide for A whom to trust, i.e. $A \overset{trust}{\Longrightarrow} B \wedge B \overset{trust}{\Longrightarrow} C = A \overset{trust}{\Longrightarrow} C$. Another property of a trust relation is that it is asymmetric [18, 26], i.e. if A trusts B in proposition x then nothing may be said about B's trust on A in x. In addition to logical properties, a trust relation needs to allow controversial experiences.

Each viable system implementing trust relations needs to consider the representation of trust and the means to compute with it. The representation can be binary / discrete, range or continuous values and the computation therefore, logical / fuzzy, probabilistic or basing on transitiveness respectively. Existing implementations to compute with these representations include, but are not limited to, summation [27, 28], fuzzy models [29], Bpdf [9, 30, 31, 32], EigenTrust [33] and PageRank [34]. The scale of a trust metric can be of any kind. It however needs to be partially ordered and is often totally ordered, e.g. $\mathbb{R}$ [0, 1] with relation $\leq$, {-1, 0, +1} with $-1 \leq 0 \leq 1$, {low, mediocre, somewhat, high} where low $\leq$ mediocre $\leq$ high and low $\leq$ somewhat $\leq$ high. Hence, the interpretation of the outcome may be a threshold, rank, probability or mere cognition leaving it up to the human to decide.

2.2. Foundations of the Three-Valued parameter

This paper considers trust to be a three-valued metric. The metric bases on Belief functions, or Dempster-Shafer theory, that is a generalisation of Bayesian theory of subjective probability. A Belief function operates on a set of known outcomes X where the mass (certainty) m: $2^X \rightarrow$ [0, 1] denotes the evidence of each outcome. The probabilistic view on the evidence assigns m to

each element 2^X and is called basic belief assignment where $m(\emptyset) = 0$ and $\sum_{A \in 2^X} m(A) = 1$. This additivity is modelled on a mass space, e.g. $X = \{x_1, x_2, x_3\}$ where the mass 'x_1 or x_2' denote the certainty of not x_3, but not certain whether x_1 or x_2, i.e. the mass of $(\{x_1, x_2\})$. Realistically this is the case when X denotes colours of balls in a box, say $\{red, green, black\}$ and the evaluator is red-green colour blind knowing that the ball drawn was not black.

In addition to the mass m, belief bel is defined $bel(A) = \sum_{B \subseteq A} m(B)$. Hence, bel denotes the 'certainty' or 'evidence' in a set of interest as the sum of masses that are subsets of it, e.g. $bel(\{red, green\}) = m(\{red\}) + m(\{green\}) + m(\{red, green\})$. The mass of the total set $m(X)$ need not be 0, i.e. $m(\{red, green, black\}) \neq 0$. Plausibility pl denotes the 'max probability' or that 'there is evidence against this proposition' where $pl \geq bel$ and $pl(A) = \sum_{A \cap B \neq \emptyset} m(B)$, the sum of non empty intersecting masses; or more conveniently, $pl(A) = 1 - bel(\bar{A})$ where $\bar{A}$ denotes complement of A, in this case $1 - bel(\{black\})$.

With mass, belief and plausibility provides the upper (pl) and lower (bel) bounds of probability. This interval between pl and bel is the uncertainty, the scope of lacking evidence in favour for or against the set of interest constituting the third-value in our trust metric.

2.3. Representation of Experiences

To represent the levels of trust, we propose a general representational model for experience based trust relying on the history of recorded experience(s) on subjects in propositions. The model follows Krukow's general model [35]. In this model, the history of experiences is defined by a set of 4-tuples $Exp^{trustor}(\epsilon_0) = \{(\delta, \epsilon, \zeta, \eta)\}$ where *trustor* is the software agent whose experiences are examined, $\delta \in subject$ is the trustee's long term identification with whom the experience was, ϵ is the datum where $\epsilon \leq \epsilon_0$, $\zeta \in proposition$ and $\eta \in score$. The datum may be virtually any continuous matter, typically time. For example, $(Bob, \epsilon_0, x, \eta) \in Exp^{Alice}$ denotes that at datum ϵ_0 *Alice* recorded an experience on Bob in proposition x with score η where *Bob* may provide x or act as a referral to another δ providing for x, the transitivity. Moreover, we write $Exp_\delta^{\delta'}(\epsilon)$ for the δ-selection on the history of δ'. Then sel_ϕ lists all rows that satisfy some predicate ϕ. E.g. $Exp_{Bob}^{Alice}(\epsilon)$ provides a set of n-tuples $\{(\epsilon_n, \zeta, \eta)\}$ where $\phi \triangleq \delta = Bob$ and dually, $Exp_{Bob}^{Alice}(\epsilon, food)$ where $\phi \triangleq \delta = Bob \wedge \zeta = food$ provides a set of n-tuples $\{(\epsilon_n, \eta)\}$. With this syntax and selections, $Exp_\delta^{\delta'}(\epsilon_0) \subseteq Exp^{\delta'}(\epsilon_0)andExp_\delta^{\delta'}(\epsilon_i) \subseteq Exp^{\delta'}(\epsilon_0)$ when $\epsilon_i \leq \epsilon_0$.

2.3.1. Experience Type

Having defined the general model for representing experiences, we consider the type of each such experience a tuple (*sat*, *unsat*), i.e. $Exp_{Bob}^{Alice}(\epsilon_i, \zeta) = \{(sat, unsat)\}$, the η-projection on sel_ϕ. In this tuple, *sat* and *unsat* denotes the level of satisfactory and unsatisfactory behaviour respectively. Characteristics include that *sat*, *unsat* ∈ [0, 1] and *sat* + *unsat* ≤ 1, i.e. the tuple may be subadditive. Subadditivity is fundamental for decay, described in Section 2.3.2. This type of experience allows complete uncertainty to be expressed as (0, 0), i.e. no evidence of either satisfactory or unsatisfactory behaviour, dogmatic experiences as (*sat*, *unsat*) where *sat* + *unsat* = 1 and absolute experiences when (*sat*, *unsat*) = (0, 1) or (1, 0). Moreover, the type enables simple aggregation of experience by summation on η-projection of a selection on a subject in proposition with an initial view at ϵ_i as $\sum Exp_{Bob}^{Alice}(\epsilon_i, \zeta).sat = 0$ and $\sum Exp_{Bob}^{Alice}(\epsilon_i, \zeta).unsat = 0$; that is, a view of no evidence.

2.3.2. Experience Decaying and Abstracting Experiences

Each agent's experiences at ϵ is defined $Exp^{\delta'}(\epsilon) = \{(\delta, \epsilon, \zeta, \eta)\}$ making the set of experiences at m defined by $Exp^{\delta'}(\epsilon_m) = Exp^{\delta}(\epsilon_{m-1}) \cup \{(\delta, \epsilon_m, \zeta, \eta)\}$. On these experiences, the decaying of the relative weigh of each experience with respect to ϵ is necessary for rapidly adjusting to changes in the autonomous subject's behaviour. The method of decay must recognise the independence of the three-valued parameter metrics, i.e. it must not subvert the experience, merely reduce its weigh.

Let the decay factor be λ defined $0 \leq \lambda \leq 1$ on a continuous datum ϵ. This defines the general decay function d at ϵ_n called d_{ϵ_n}as:

$$d_{\epsilon_n}\left(Exp^{\delta}(\epsilon_0)\right) = \{(\delta, \epsilon, \zeta, \lambda^{\epsilon_n - \epsilon} * \eta)\}$$

Where each experience is decayed by λ defining the 'forgetting' speed where the closer to 1, the less speed and trivially, λ = 1 is no decay whereas λ = 0 is complete [36]. Complete decay is motivated when aprior experiences may not be used to estimate posterior outcomes, e.g. in case of idealised lottery. Hence, the affect of decay is that an experience score η is reduced by factor λ on datum, i.e. η at $\epsilon_n \leq \eta$ at ϵ_m when $n \leq m$ whenever λ < 1. Realistically, if ϵ is time and λ < 1, then experiences are decayed by time.

The abstracted experience is a composition of the disjoint experiences on the subject in a proposition. Having decayed experiences, the abstracted decayed experiences provide a tuple $(absDsat, absDunsat)$ as the η-projection on the decayed selection, defined:

$$Abs_{\epsilon_n}\left(Exp_{\delta}^{\delta'}(\epsilon, \zeta)\right) = \sum d_{\epsilon_n} Exp_{\delta}^{\delta'}(\epsilon_n, \zeta)$$

That is, as $d_{\epsilon_n} Exp_{\delta}^{\delta'}(\epsilon_n, \zeta)$ provides a set $\{\epsilon, (absSat, absunsat)\}$ and the η-projection restricts this to $\{(absSat, absunsat)\}$, then the sum on this is called $(absDsat, absDunsat)$.

Not surprisingly, as $Abs_{\epsilon_n}\left(Exp_{\delta}^{\delta'}(\epsilon, \zeta)\right)$ denotes the tuple decayed on datum ϵ_n, an updated abstract view $Abs_{\epsilon_m}\left(Exp_{\delta}^{\delta'}(\epsilon, \zeta)\right)$ where $m \geq n$ is a recursive function whenever the decaying factor is universal, continuous and applied on all experiences locally, e.g. decay by time. Hence, updating $Abs_{\epsilon_n}\left(Exp_{\delta}^{\delta'}(\epsilon, \zeta)\right)$ is straight forward.

$$Abs_{\epsilon_m}\left(Exp_{\delta}^{\delta'}(\epsilon, \zeta)\right) = Abs_{\epsilon_{m-1}}\left(Exp_{\delta}^{\delta'}(\epsilon, \zeta)\right) * \lambda + \eta$$

Here, η = (*sat*, *unsat*) at time ϵ_m, i.e. the new experience. Thereby, abstraction is an irreversible function that provides a level of privacy that decay enhances on. When no experience occurred at time ϵ_m, $Exp_{\delta}^{\delta'}(\epsilon_m, \zeta) = (0, 0)$. Moreover, $\sum d_{\epsilon_n} Exp_{\delta}^{\delta'}(\epsilon_n, \zeta) = \sum Exp_{\delta}^{\delta'}(\epsilon_n, \zeta)$ if $\lambda = 1$, i.e. no decay.

2.4. Trust Networks and Trust Transitivity

The basis of trust evaluation where the trustor (*s*) derives a level of trust in a trustee (*t*) may be considered as a graph $G = (V, E)$ where V is a set of vertices and edges E a set of ordered pairs of vertices. Such a graph, hereafter network, may expand by adding intermediary referral nodes. In a Bayesian network this is defined a directed acyclic graph (DAG). However, when instead of binary 'AND' and 'OR', probabilistic multiplication and co-multiplication on incomplete opinions are used for serial and parallel composition, the DAG does not qualify [25, 37]. The problem is coined to parallel and serial path confusion, e.g. let $\{(S, A), (A, B), (A, C), (S, B), (B, C), (C, D)\} \subseteq E$, then either (A, B) or (A, C) is to be discarded or independence is violated.

The solution is to limit the DAG to a Series Parallel Graph (SPG). A SPG may be constructed by applying the following series and parallel rules on a graph $G_2 = (\{S_1, S_2, S_3\}, \{(S_1, S_2)\})$ [38]:

Series: replace the edge (S_1, S_2) with (S_1, S_3) and (S_3, S_2) where S_3 is a new vertex

Parallel: replace the edge (S_1, S_2) with two edges $(S_1, S_2)_1$ and $(S_1, S_2)_2$

Hence, in a SPG either edge (A, B) or (A, C) may not be created. Moreover, as a trust relation is directed, the SPG becomes a Directed SPG (DSPG). Each DSPG is a DAG. Hence, the DSPG outlines trust transitivity. Moreover, with edge splitting as proposed in [39], a DAG may be transformed into a DSPG assuming a "fission factor" ψ_i where $\sum_{i=1}^{n} \psi_i = 1$ on which path of the i paths to take. This is similar to defining the probability of selecting (A, B) over (A, C); preserving the analytical possibilities of such a network. Moreover, the perceived topology of a network must concur with the real topology, i.e. no edge may occur twice. Interested readers are directed elsewhere [25].

3. Calculating with Experiences

The type of a score η of an experience is defined a tuple (*sat*, *unsat*). Abstractions of experiences include tuple (*abssat*, *absunsat*) and decayed tuple (*absDsat*, *absDunsat*). To calculate with these capturing uncerainty in a structure alike a DSPG, functions on parallel and sequential composition on a three-valued metric need to be defined. Moreover, when composing several DSPGs, functions for multiplication and co-multiplication is demanded. For this, SL fits

well. SL is a probabilistic logic basing on belief theory that takes uncertainty and the trustor into account [9, 11, 12, 22]. The SL defines an opinion in the interval [0, 1]. Hence, it is related to the B-family of probability density functions (Bpdf) and Dirichlet pdf for *k*-dimensions. Moreover, it may be used to Bayesian networks as conditional reasoning functions have been defined, interested readers are directed to referenced literature [40, 41, 42]. The SL must not be confused with fuzzy logic as the latter operates on crisp and certain measures about linguistically vague and fuzzy propositions; whereas SL operates on uncertain measures about 'crisp' propositions [11].

The level of (un)trustworthiness in SL is defined by (*dis*)*belief* and (*un*)*certainty* on a subject in proposition, called an opinion denoted ω. The opinion is uniquely mapped from the score tuple. Thereby, it builds up and changes by datum and decay operator. Moreover, it is a generalisation of binary logic, i.e. whenever an SL opinion is absolute, the SL functions behave alike their corresponding logical expressions [12]. In addition, the level of trustworthiness perceived in a software agent varies, stressing the impossibility of defining a "globally correct behaviour". This implies impossibility of applying formal approaches extensively as no precise assumptions on the environment of the (formal) model may be taken [43]. The following subsections elaborating on means to calculate with an opinion base on work by Jøsang et al. [9, 10, 11, 12].

3.1. Foundations for an Opinion

An opinion is a three-valued metric on a certain outcome of possible outcomes. To explain the opinion, consider a set of exclusive and exhaustive outcomes $\mathcal{X}$, called a frame, e.g. $\mathcal{X} = \{x_1, x_2, x_3\}$. An opinion on the frame is defined as a 3-tuple $(\vec{b}, u, \vec{a}\,)$ of a *b*elief mass vector, *u*ncertainty mass scalar and base rate vector *a* in a *k*-nomial barycentric coordinate system where $k = |\mathcal{X}|$. The vectors $\vec{b}$ and $\vec{a}$ are vector-valued functions on the propositions of $\mathcal{X}$ with range $[0, 1]^k$, e.g. $(\vec{b}(x_1), u, \vec{a}(x_1))$, $(\vec{b}(x_2), u, \vec{a}(x_2))$, $(\vec{b}(x_3), u, \vec{a}(x_3))$ as of trinomial $\mathcal{X}$. The beliefs are subadditive, i.e. $\sum_{x_i \in \mathcal{X}} \vec{b}(x_i) \leq 1$ with the uncertainty u covering for additivity, $u = 1 - \sum_{x \in \mathcal{X}} \vec{b}(x)$, i.e. $u \in [0, 1]$; whereas the base rate vectors are additive $\sum_{x \in \mathcal{X}} \vec{a}(x) = 1$. Hence, the length of $\vec{b}(x_i)$ denotes the evidence, *bel* in belief functions on a proposition x_i.

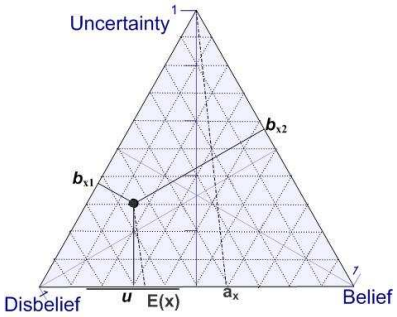

Figure 2. A binomial opinion triangle

A multinomial frame $|\mathcal{X}| \geq 3$ may be coarsened to a binomial frame when partitioned into x and its complement $\bar{x}$, i.e. $\mathcal{X}' = \{x, \bar{x}\}$ and $|\mathcal{X}'| = 2$. Hence, we will hereafter consider $|\mathcal{X}'| = 2$, directing readers interested in $|\mathcal{X}| \geq 3$ to [11]. The binomial form of an opinion is $\vec{b}(x), \vec{b}(\bar{x}),\ u, \vec{a}(x)$ and may be illustrated as a point in the binomial barycentric coordinate system **Figure 2**. This is the opinion defined as a four tuple $\omega_x = (\vec{b}(x), \vec{b}(\bar{x}),\ u, \vec{a}(x))$ where belief $b = \vec{b}(x)$, disbelief $d = \vec{b}(\bar{x})$, uncertainty $u = \{x, \bar{x}\}$ and base rate a is given. Moreover, $b + d + u = 1$.

As of the changing environment, we consider all opinions with a level of uncertainty, $u > 0$. Hence, the base rate vector a comes to be decisive in finding the expectation value. The expectation value of an opinion on a proposition x is defined $E(\omega_x) = b + u * a$ denoting the posterior belief mended by the uncertainty. In **Figure 2** this is the interval on the base spanned by orthogonal vectors to $\vec{b}(x)$ and $\vec{b}(\bar{x})$. Obviously, the posterior belief satisfies additivity, $\sum_{x \in \mathcal{X}} E(\omega_x) = 1$, hence establishing a crisp value of expected probability for each proposition of $\mathcal{X}$ [10].

The expectation value proves its importance when ordering opinions in a total order based on belief. Otherwise, deciding whether $\omega_x \leq \omega_z$ or $\omega_x \geq \omega_z$ for arbitrary propositions x and z for example when $\omega_x = (0.3, 0.3, 0.4, a)$ and $\omega_z = (0.4, 0.4, 0.2, a)$ is impossible as ω_z depict more trust, but more distrust as well. With respect to belief theory, the point in the barycentric coordinate system is where *belief* = $\sum_{x \in \mathcal{X}} \vec{b}(x)$ and *disbelief* = $\sum_{\bar{x} \in \mathcal{X}} \vec{b}(\bar{x})$ intersect defining u (gap between *bel* and *pl*).

3.2. Mapping from Abstracted Experiences to an Opinion

Having defined the experiences' score as a tuple η and as the SL apply on opinions, a mapping function is desired. Consider a η–projection on a selection of $Abs_{\epsilon_n}\left(Exp_{\delta}^{\delta'}(\epsilon, \zeta)\right)$ providing the (*absDsat*, *absDunsat*). This tuple may be converted to and from an opinion ω by the mapping relation (1) originally proposed by Jøsang [9] and later elaborated on in [11, 25, 32]:

$$\omega \begin{cases} b = \frac{absDsat}{absDsat+absDunsat+W} \\ d = \frac{absDunsat}{absDsat+absDunsat+W} \\ u = \frac{W}{absDsat+absDunsat+W} \\ a = base\ rate \end{cases} \Leftrightarrow \begin{cases} absDsat = \frac{Wb}{u} \\ absDunsat = \frac{Wd}{u} \\ a = base\ rate \end{cases} tuple \tag{1}$$

In this mapping, the parameter W denotes the non-informative prior weight. It also guarantees $u > 0$. When (*absDsat*, *absDunsat*) = (0, 0), $u = 1$. The Bpdf input parameters $\alpha = absDsat + Wa$ and $\beta = absDunsat + Wa$ that indicate a uniform distribution whenever $a = 0.5$. Bpdf:s may be used to illustrate an opinion. Greater W slows the influence of evidence [11, 25]. This mapping relation is central as experiences' score η are recorded as a tuple and calculations on them are done as opinions.

3.3. Functions of Subjective Logic

In order to calculate on opinions, some functions are demanded. To provide the functions, consider a subject δ to direct experiences in propositions $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, i.e. (δ, x) by η–projection on a selection $Abs_{\epsilon}\left(Exp^{\delta}(\epsilon, x)\right)$ and (δ, y) by $Abs_{\epsilon}\left(Exp^{\delta}(\epsilon, y)\right)$. Mapping these to opinions by (1) provides $\omega_x^{\delta} = (b_x, d_x, u_x, a_x)$ and $\omega_y^{\delta} = (b_y, d_y, u_y, a_y)$. The opinion ω is labelled by the source as upper and target as lower index. An opinion $\omega_{x \wedge y}^{\delta}$ indicates a multiplication of two propositions and $\omega_{x \vee y}^{\delta}$ co-multiplication by δ's opinions. Moreover, we use ';' for sequential and $\diamond$ for parallel composition, e.g. $\omega_x^{\delta;\, \delta'} = \omega_{\delta'}^{\delta}$; $\omega_x^{\delta'}$ and $\omega_x^{\delta \diamond\, \delta'} = \omega_x^{\delta} \diamond \omega_x^{\delta'}$.

Multiplication is the function for the opinion on outcome of $\{(x, y)\} \in \mathcal{X} \times \mathcal{Y}$, written $\omega_x^{\delta} \wedge \omega_y^{\delta} = \omega_{x \wedge y}^{\delta} = \left(b_{x \wedge y}, d_{x \wedge y}, u_{x \wedge y}, a_{x \wedge y}\right)$. It is defined following [37]:

$$\omega_{x \wedge y}^{\delta} = \begin{cases} b_{x \wedge y} = b_x b_y + \frac{(1-a_x)a_y b_x u_y + (1-a_y)a_x b_y u_x}{1-a_x a_y} \\ d_{x \wedge y} = d_x + d_y - d_x d_y \\ u_{x \wedge y} = u_x u_y + \frac{(1-a_y)b_x u_y + (1-a_x)b_y u_x}{1-a_x a_y} \\ a_{x \wedge y} = a_x a_y \end{cases} \tag{2}$$

Having the same propositions, co-multiplication denotes the opinion on outcomes $\{(x, y), (x, \bar{y}), (\bar{x}, y)\} \in \mathcal{X} \times \mathcal{Y}$, written $\omega_x^{\delta} \vee \omega_y^{\delta} = \omega_{x \vee y}^{\delta} = \left(b_{x \vee y}, d_{x \vee y}, u_{x \vee y}, a_{x \vee y}\right)$. It is defined following [37]:

$$\omega_{x \vee y}^{\delta} = \begin{cases} b_{x \vee y} = b_x + b_y - b_x b_y \\ d_{x \vee y} = d_x d_y + \frac{(1-a_y)a_x d_x u_y + (1-a_x)a_y u_x d_y}{a_x + a_y - a_x a_y} \\ u_{x \vee y} = u_x u_y + \frac{a_y d_x u_y + a_x d_y u_x}{a_x + a_y - a_x a_y} \\ a_{x \vee y} = a_x + a_y - a_x a_y \end{cases} \tag{3}$$

Multiplication and co-multiplication are commutative but not distributive, e.g. $\omega^{\delta}_{x\wedge y} = \omega^{\delta}_{y\wedge x}$ but $\omega^{\delta}_{x\wedge(y\vee z)} \neq \omega^{\delta}_{x\wedge y} \vee \omega^{\delta}_{x\wedge z}$. Thereby, multiplication and co-multiplication are the functions for composing two exclusive propositions of disjoint frames. These are well formed with the exception of multiplication when $a_x = 1$ and $a_y = 1$, and for co-multiplication except for when $a_x = 0$ and $a_y = 0$.

With respect to probabilistic calculations, calculation of belief in multiplication and disbelief in co-multiplication deviates. This is to get the expectation value to converge with its probabilistic peer and keeping the base rate motivated [37]. For example, consider ω^{δ}_{x} = (0.466, 0.074, 0.459, 0.5) with $E(\omega^{\delta}_{x}) = 0.696$ and ω^{δ}_{y} = (0, 0.685, 0.313, 0.5) with $E(\omega^{\delta}_{y}) = 0.158$, for $E(\omega^{\delta}_{x} \wedge \omega^{\delta}_{y}) = E(\omega^{\delta}_{x}) * E(\omega^{\delta}_{y})$, this deviation is necessary.

Deriving an opinion in a target from multiple paths of a DSPG is called consensus and discounting. Consider a DSPG G' = ({*S*, *X*, *Y*, *Z*, *t*}, {(*S*, *X*), (S, *Y*), (*X*, *Z*), (*Y*, *Z*), (*Z*, *t*)}) where vertex *Z* has direct functional trust in an arbitrary proposition *t*. The two paths are ρ_1 = (*S*, *t*) = (*S*, *X*) ; (*X*, *Z*) ; (*Z*, *t*) and ρ_2 = (*S*, *t*) = (*S*, *Y*) ; (*X*, *Z*) ; (*Z*, *t*) that combined is (((*S*, *X*) ; (*X*, *Z*)) ⋄ ((*S*, *Y*) ; (*X*, *Z*)) ; (*Z*, *t*). To calculate the opinion from these paths consensus and discounting are needed. Discounting denoted ⊗ operates on serialised opinions denoted ';', i.e. $\omega_t^{(S;X;Z)} = \omega_X^S \otimes \omega_Z^X \otimes \omega_t^Z$. By discounting software agent *X* evidence in *Z* is related by *S*'s evidence in *X* and there exist at least three different means for discounting an opinion $\omega_t^{S;X}$ [44]:

$$i)\begin{cases} b_t^{S;X} = b_X^S b_t^X \\ d_t^{S;X} = b_X^S d_t^X \\ u_t^{S;X} = d_X^S + u_X^S + b_X^S u_t^X \\ a_t^{S;X} = a_t^X \end{cases}, ii)\begin{cases} b_t^{S;X} = b_X^S b_t^X + d_X^S d_t^X \\ d_t^{S;X} = b_X^S d_t^X + d_X^S b_t^X \\ u_t^{S;X} = u_X^S + (b_X^S + d_X^S) u_t^X \\ a_t^{S;X} = a_t^X \end{cases}, iii)\begin{cases} b_t^{S;X} = E(\omega_X^S) b_t^X \\ d_t^{S;X} = E(\omega_X^S) d_t^X \\ u_t^{S;X} = 1 + E(\omega_X^S) u_t^X - E(\omega_X^S) \\ a_t^{S;X} = a_t^X \end{cases} \quad (3)$$

Case (i) is discounting while favouring uncertainty, originally proposed in [9]. Case (ii) view conflicting opinions as belief, i.e. *your enemy's enemy is your friend* [25]. For case (ii), the authors [25] note that modelling chains longer than two edges with this methodology is doubtful. The third case (iii) operates on expectation values being a bad choice at high uncertainty, but might in special cases be the least bad choice, called base rate sensitive discounting. In case (iii), expectation $E(\omega_X^S) = b_X^S + (u_X^S a_X^S)$, as before. Discounting is trivially asymmetric.

Contrary to discounting, consensus ⊕ enforces the evidence in a third party by combining parallel paths. Consensus is denoted ⊕ and is the operation of combining parallel opinions denoted ⋄, i.e. $\omega_t^{((S;X);(X;Z))\diamond((S;Y);(Y;Z))} = \omega_t^{((S;X);(X;Z))} \oplus \omega_t^{((S;Y);(Y;Z))}$. Hence, ω_t^S of DSGP G' is $\rho_1 \diamond \rho_2$ which by opinions is $\omega_t^S = \omega_t^{(((S,X);(X,Z))\diamond((S,Y);(Y,Z)))} = \left((\omega_X^S \otimes \omega_Z^X) \oplus (\omega_Y^S \otimes \omega_Z^Y)\right) \otimes \omega_t^Z$. The first variant of consensus was proposed in [9] whereas only later, the consideration of a priori base rate *a* was included [11], defined:

$$\begin{aligned} b_t^{(S;X)\diamond(S;Y)} &= \left(b_t^{(S;X} u_t^{(S;Y)} + b_t^{(S;Y)} u_t^{(S;X)}\right) / \left(u_t^{(S;X)} + u_t^{(S;Y)} - u_t^{(S;X)} u_t^{(S;Y)}\right) \\ d_t^{(S;X)\diamond(S;Y)} &= \left(d_t^{(S;X)} u_t^{(S;Y)} + d_t^{(S;Y)} u_t^{(S;X)}\right) / \left(u_t^{(S;X)} + u_t^{(S;Y)} - u_t^{(S;X)} u_t^{(S;Y)}\right) \\ u_t^{(S;X)\diamond(S;Y)} &= \left(u_t^{(S;X)} u_t^{(S;Y)}\right) / \left(u_t^{(S;X)} + u_t^{(S;Y)} - u_t^{(S;X)} u_t^{(S;Y)}\right) \\ a_t^{(S;X)\diamond(S;Y)} &= \frac{a_t^{(S;Y)} u_t^{(S;X)} + a_t^{(S;X)} u_t^{(S;Y)} - \left(a_t^{(S;X)} + a_t^{(S;Y)}\right) u_t^{(S;X)} u_t^{(S;Y)}}{u_t^{(S;X)} + u_t^{(S;Y)} - 2 u_t^{(S;X)} u_t^{(S;Y)}} \end{aligned} \quad (4)$$

With these fundamental functions on opinions, it is possible to calculate the possible structures of a DSPG as well as combine disjoint DSPGs.

4. The Notion of a Group by Trustworthiness

Trustworthiness relations have been identified among others as one-to-many or many-to-one [18]. Here, the 'many' captures the concept of a set of actors. We consider this 'many' a group in the context of deriving a level of trust on a subject in proposition. A group is a set of software agents that are categorised by a bond by likes. This notion of a group lends itself from social sciences peer group, where the social background, roles, statuses are excluded and members interact possibly on the level of a group, share a common goal and are bonded by the

likes. Examples of real-life peer groups are friends, fan club and community. Related work on groups in a similar context includes [29]. They do however choose the most representative agent from a set, as a kind of supernode that represents the "witness" merely to reduce the number of queries, not to categorise by trustworthiness. Hence, the way we treat a group is different.

This Section defines such a group, where the likeness is defined by the relative frequency of the decayed abstracted experiences on some proposition(s). Moreover, this Section outlines the implications of such a group on the software agents and presents how these may be utilised to ascertain a level of trust by trustworthy experiences in a subject in a proposition. To the best of our knowledge, the presented approach is novel.

4.1. Definition of a Group

A group is a set of agents that are bonded by the similarity of their experiences on a subject in a proposition. We consider such a group a software agent in its own right. Thereby, any agent referring to a group is provided the group's aggregated experiences. As the group abstracts a set of software agent(s) and all software agents have complete trust in themselves, the group has complete trust in its members. Hence, let Y, $Z \in$ subject and $x \in$ proposition and tuple (low, $high$) denote the thresholds for grp, then a group $G_\epsilon(Z_x^{grp})$ is defined:

Definition of a Group:

$$\begin{cases} if\ Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,x)\right).absDunsat \neq 0 \\ \qquad then\ G_\epsilon\left(Z_x^{grp}\right) = \left\{Y \in \delta : low^{grp} \leq \frac{Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,x)\right).absDsat}{Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,x)\right).absDunsat} \leq high^{grp}\right\} \\ else\ if\ Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,x)\right).absDunsat = 0 \wedge Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,x)\right).absDsat \neq 0 \\ \qquad then\ G_\epsilon\left(Z_x^{grp}\right) = \{Y \in \delta : low^{grp} \leq max \leq high^{grp}\} \end{cases}$$

Hence, the group $G_\epsilon\left(Z_x^{grp}\right)$ is a software agent abstracting a set of software agents that share the likes on subject Z in a proposition x at time ϵ constrained by thresholds low^{grp} and $high^{grp}$. Realistically, let Z be a restaurant and x food taste, then $G_\epsilon\left(Z_x^{grp}\right)$ is a set of software agents who share likes on Z in x. Whenever both $absDsat = 0$ and $absDunsat = 0$, an agent is vacuous with respect to the proposition and may not belong to any group over such.

The group's $G_\epsilon(Z_x^{grp})$ experience on a proposition x is defined by summation:

$$\left(\sum_{Y \in G_\epsilon(Z_x^{grp}) \setminus \text{inquirer}} Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,x)\right)\right) \qquad (5)$$

Hence, the group's experience is the sum of its members' experiences excluding the inquirer. On this, two observations may be made: firstly, the group's experiences depend on who inquires and secondly, a group that the inquirer does not share likes with may be inquired. The summation of ($absDsat$, $absDunsat$) is equivalent to consensus on the disjoint members' opinions in the proposition where discounting is excluded as of complete trust.

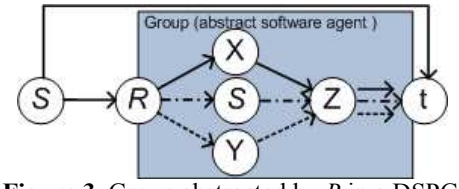

Figure 3. Group abstracted by R in a DSPG

This definition of a group has four features. Firstly (i), as the members of a group are dynamic, the trustor perceives experiences in the group software agent, R in **Figure 3**. We consider for brevity experience to be evenly distributed, i.e. $Exp^S(\epsilon_0) = Exp^S(\epsilon_1) \cup \left\{\delta_i, \epsilon_0, \zeta, \left(\frac{sat}{deg^+(S)}, \frac{unsat}{deg^+(S)}\right)\right\}$ where deg^+ is the outdegree edges from S and $i = 1$, deg^+. Secondly (ii), as the trustor inquires referrals for their experiences, referring to itself is unmotivated. Hence, the trustor is excluded from the group. This implies that referring to a singleton group whose member is the inquirer provides (0, 0), i.e. in accordance to $\sum_\emptyset x = 0$. Thirdly (iii), as a

group is defined on a subject in a proposition, it abstracts direct functional trust relations, i.e. a relation on the proposition. With respect to **Figure 3**, software agent $G_\epsilon(R_t^{grp})$ where R is a subject abstracts (((*R*; *X*);(*X*; *Z*)) ⋄ ((*R*; *S*); (*S*; *Z*)) ⋄ ((*R*; *Y*);(*Y*; *Z*))); (*Z*; *t*). As of *R*'s complete trust in its members and discounting as well as *S* being the trustor, $G_\epsilon(R_t^{grp})$ collapses to ((*X*; *Z*) ⋄ (*Y*; *Z*)); (*Z*; *t*). The real topology of the opinion ω_t^S in structure DSPG of **Figure 3** concurs with the perceived and is therefore $\omega_t^S \oplus \left(\omega_R^S \otimes \omega_t^{G_\epsilon(R_t^{grp})}\right)$. The last feature (iv) is that the thresholds define the group(s) and are decisive. These thresholds need to be exclusive and exhaustive, i.e. any software agent $\in G_\epsilon(Z_x)$ is a member of one $G_\epsilon(Z_x^{grp_i})$. That is, $\bigcap_i G_\epsilon(Z_x^{grp_i}) = \emptyset$ and $\bigcup_i G_\epsilon(Z_x^{grp_i}) = G_\epsilon(Z_x)$ for $i = 1 \ldots n$.

4.2. Setting the Thresholds

The threshold values *low*, *high* ∈ [0, 1] restricting the group are defined as a sorted ≤ array (set of tuples) *Ar* of ℝ, e.g. $Ar[a, b]$ where $a \leq b$. However, for exhaustiveness the smallest value and thresholds inverses are included. Therefore, the array *Ar* expands to *Ar*'$[0, a, b, \bar{b}, \bar{a}, \bar{0}]$ where $0 \leq a \leq b \leq \bar{b} \leq \bar{a} \leq \bar{0}$ with inverses defined $\overline{val} = 1/val$ and $\bar{0} = max$ that in case of ℝ = ∞. The number of groups in a view on a subject in proposition is always odd and is (*Ar*'.*length*-2) + 1 because excluding 0 and $\bar{0}$ defining the intervals.

An empty array *Ar* gives rise to a global group; all non-vacuous agents belong to this group. The group order on a software agent $A \in G_i$ for $i = 0, \ldots, k$ where k = (*A*'.*length*-2) + 1 is $Ar'[0] \leq G_1 \leq Ar'[1] \leq G_2 \leq Ar'[3] \leq \cdots \leq Ar'[k-2] \leq G_4 \leq Ar'[k-1] \leq G_5 \leq Ar'[k]$. Hence, consider an agent with *m* = *absDsat* and *n* = *absDunsat*, then if $(m, n) \in G_v$ the experience $(n, m) \in G_w$ where $w = k - v + 1$, e.g. with $k = 5$ and $(m, n) \in G_2$ then $(n, m) \in G_4$.

4.3. Group Customisation and Composition

Group customisation refers to composition of groups by a set theoretic operator *op*. A customised group is hence an abstraction of its underlying software agents. Let *X* be a set of groups on subjects in propositions, $X = \left\{G_\epsilon(Z_{1\zeta_1}^{grp_1}), G_\epsilon(Z_{2\zeta_2}^{grp_2}), \ldots, G_\epsilon(Z_{n\zeta_n}^{grp_n})\right\}$. Then, the customised group is defined:

$$G_\epsilon^{op}(Z_\zeta^{grp}) = \left\{op_{Z_{\zeta_i}^{grp_i} \in X} G_\epsilon(Z_{\zeta_i}^{grp_i})\right\}$$

The *op* of intersection on groups *Z* bonded by ζ_i provides the most restrictive configuration where all members of $G_\epsilon^{\cap_i}(Z_\zeta^{grp})$ adheres to all propositions. Contrary the *op* of union on ζ_i is the most liberal group composition.

Other logical combinations of groups in disjoint propositions are also possible. Realistically, consider a set of restaurants $\{Z_1, Z_2, \ldots, Z_n\}$ and propositions on these $\zeta = \{ft, p, at\}$ for food taste, placing and atmosphere respectively, then the groups are $\left\{G_\epsilon^{op}\left(Z_{j\,ft}^{grp}\right), G_\epsilon^{op}\left(Z_{j\,p}^{grp}\right), G_\epsilon^{op}\left(Z_{j\,at}^{grp}\right)\right\}$ for $j = 1, \ldots, n$. Hence. for example, *Bob* inquiring for agents that bond by his likes in restaurant(s) Z_j by food taste and either placing or atmosphere is defined:

$$G_\epsilon^{ft\cap(p\cup at)}(Z_\zeta^{grp}) = \left\{Bob \in Z_{\zeta_i}^{grp_i} : \left(G_\epsilon\left(Z_{j\,ft}^{grp}\right) \cap \left(G_\epsilon\left(Z_{j\,p}^{grp}\right) \cup G_\epsilon\left(Z_{j\,at}^{grp}\right)\right)\right)\right\}$$

Hence, group customisation defines by set theory the referrals as members of the group that is considered trustworthy. Extending this to model groups ascertaining by related subjects in a proposition is straight forward. For example, consider restaurant *R* to be related to $Z = \{Z_1, Z_2, \ldots, Z_n\}$, realistically they may all serve Italian food. Then, ascertaining experiences of *R* by the experiences of the group sharing likes on *Z* is reasonable, i.e. $G_\epsilon(R'^{grp}_\zeta) = G_\epsilon(R_\zeta^{grp}) \cup G_\epsilon(Z_\zeta^{grp})$ that by (5) provides the members' summed experiences on *R*. Consequently, the model enables a previously uncertain subject in proposition be ascertained also by related subjects in proposition.

5. Case Restaurant Evaluation System

To exemplify the contribution of this paper, this section outlines an example of a restaurant evaluation system. The system itself is an application that manages multiple users' experiences in multiple subjects R_i in multiple propositions ζ manifesting the users' profiles. These are outlined in table 1, where $R_i\zeta$ stands for subject R_i in proposition ζ. The example comprises of restaurants $\{R_1, R_2\}$ with $\zeta = \{t, p, at\}$ for taste, placing and atmosphere respectively. The threshold values are *ThrA* = [3/10, 3/5] whereas *sat* denotes *absDsat* and *unsat absDunsat*, *b*, *d*, *u* an opinion and a *group_name* $\in$ {*NT*, *SNT*, *ST*, *VT*, *ExT*} for 'not true', 'somewhat not true', 'somewhat true', 'very true' and 'extremely true'. Let the order be $0 \leq NT \leq \frac{3}{10} \leq SNT \leq \frac{3}{5} \leq ST \leq \frac{5}{3} \leq VT \leq \frac{10}{3} \leq ExT \leq \bar{0}$. The considered software agents are *A*, *B*, *C* and *D*. The table is read so that columns indicate an agent's experiences and rows the subject and proposition. Thereby, for example agent *A* has no experience in restaurant $R_2\zeta$ and *C* have in R_2p abstract decayed experiences (1, 7) indicating bonding with a group named NT as $0 \leq \frac{1}{7} \leq \frac{3}{10}$.

Table 1: Agent experiences

what/who	A						B						C						D					
	sat	unsat	grp	b	d	u	sat	unsat	grp	b	d	u	sat	unsat	grp	b	d	u	sat	unsat	grp	b	d	u
$R_1\zeta$	10	23	SNT	0,28571	0,6571	0,0571	1	1,5	ST	0,222	0,333	0,444	11	5	VT	0,61111	0,27778	0,125	4	5	ST	0,36	0,45	0,1818
R_1t	5	6	ST	0,38462	0,4615	0,1538	0,5	0,3	ST	0,179	0,107	0,714	4	1	ExT	0,57143	0,14286	0,28571	2	2	ST	0,33	0,33	0,3333
R_1p	1	10	NT	0,07692	0,7692	0,1538	0	0,2	NT	0	0,091	0,909	3	2	ST	0,42857	0,28571	0,28571	1	2	SNT	0,2	0,4	0,4
R_1at	4	7	SNT	0,30769	0,5385	0,1538	0,5	1	SNT	0,143	0,286	0,571	4	2	VT	0,5	0,25	0,25	1	1	ST	0,25	0,25	0,5
$R_2\zeta$	0	0	null	0	0	1	6	17	SNT	0,24	0,68	0,08	6	22	NT	0,2	0,73333	0,06667	16	4	ExT	0,73	0,18	0,0909
R_2t	0	0	null	0	0	1	1	6	NT	0,111	0,667	0,222	3	6	SNT	0,27273	0,54545	0,18182	5	1	ExT	0,63	0,13	0,25
R_2p	0	0	null	0	0	1	2	3	ST	0,286	0,429	0,286	1	7	NT	0,1	0,7	0,2	6	2	ExT	0,6	0,2	0,2
R_2at	0	0	null	0	0	1	3	8	SNT	0,231	0,615	0,154	2	9	NT	0,15385	0,69231	0,15385	5	1	ExT	0,63	0,13	0,25

A decision, here buying the restaurant's service, is either affirmative or negative. It is defined by a policy that may be generalised as a predicate on the assumed posterior performance, i.e. the expectation value and certainty. For example, a predicate may be $E(\omega_x) \geq 0.8 \wedge \omega_x(u) \leq 0.1$ indicating in this case that the expected level of service a restaurant provides is at least 4 satisfactory experiences out of 5 and this to a certainty exceeding 0.9. However, when the predicate's condition on uncertainty is not fulfilled, ascertaining by referrals is motivated.

What this paper argues in favour of is that only a carefully selected subset of agents that bond by likes may qualify as referrals. Hence, assume agent *D* ascertaining its opinion in $R_1\zeta$ as a general concept. The general concept is summed or a consensus on opinions of its parts, both providing the same experience / opinion (4, 5) = (0.36, 0.5, 0.18, *a*). Then experiences of relevance for *D* in $R_1\zeta$ with *ThrA* is $G_\epsilon\left(R_{1\zeta}^{ST}\right)$, i.e.

$$G_\epsilon\left(R_{1\zeta}^{ST}\right): \left\{ Y \in \delta \wedge Abs_{\epsilon_n}\left(Exp_{R_1}^{Y}(\epsilon,\zeta)\right).absDunsat \neq 0 : \frac{3}{5}^{ST} \leq \frac{Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,\zeta)\right).absDsat}{Abs_{\epsilon_n}\left(Exp_Z^Y(\epsilon,\zeta)\right).absDunsat} \leq \frac{5}{3}^{ST} \right\}$$

providing $\{B, D\}$ whose experiences by summation (equation 5) is:

$$Abs_{\epsilon_n}\left(Exp_{R_1}^{G_\epsilon\left(R_{1\zeta}^{ST}\right)}(\epsilon,\zeta)\right) = \left(\sum_{Y \in G_\epsilon(R_{1\zeta}^{ST})} Abs_{\epsilon_n}\left(Exp_{R_1}^{Y}(\epsilon,\zeta)\right)\right) - Abs_{\epsilon_n}\left(Exp_{R_1}^{D}(\epsilon,\zeta)\right)$$

That gives (1, 1.5).

Mapping this to opinions is ≈ (0.222, 0.333, 0.444, *a*) giving rise to calculating $\omega_{R_1\zeta}^{D} = \omega_{R_1\zeta}^{D} \oplus \left(\omega_{G_\epsilon\left(R_{1\zeta}^{ST}\right)}^{D} \otimes \omega_{R_1\zeta}^{G_\epsilon\left(R_{1\zeta}^{ST}\right)}\right)$. Let us assume $\omega_{G_\epsilon\left(R_{1\zeta}^{ST}\right)}^{D}$ to be, for the sake of the example, (36, 2) = (0.9, 0.05, 0.05, *a*). With this, $\omega_{R_1\zeta}^{D} \oplus \left(\omega_{G_\epsilon\left(R_{2\zeta}^{ST}\right)}^{D} \otimes \omega_{R_2\zeta}^{G_\epsilon\left(R_{2\zeta}^{ST}\right)}\right)$ on only relevant data defined by *ThrA* is ≈ (0.366, 0.511, 0.11, *a*). With the decision predicate above, a decision may not be taken whatever base rate *a*. Relating this to not considering relevance, the bonding by

sameness would yield $\omega_{R_1\zeta}^D \oplus(\omega_A^D \otimes \omega_{R_1\zeta}^A)\oplus(\omega_B^D \otimes \omega_{R_1\zeta}^B)\oplus(\omega_C^D \otimes \omega_{1\zeta}^C)$ obviously requiring $\omega_A^D, \omega_B^D, \omega_C^D$ for calculation. The outcome is the same if ω_A^D, ω_C^D = (0, 0, 1, a) and provides otherwise a more certain but less "accurate" opinion, as consensus with any opinion (relevant or irrelevant) strengthens the certainty.

6. Discussion

This paper has presented a means to group agents by their experiences on a subject in a proposition. The effect of grouping is similar to excluding the most divergent abstracted experiences. The problem settings giving rise to this are well known with abuses in open environments known as inflation and deflation [45], i.e. unfairly positive or negative ratings [46]. Filtering such overly positive or negative approaches by the tuple (*absDsat*, *absDunsat*) have been considered in a Beta probability density function (Bpdf) by excluding a quantile of the most unfair ratings [36]. As filtering by a quantile is a viable solution, it does not abstract the experiences to groups, i.e. the experiences are agent based. This paper provides the foundation for a novel approach to this; that by categorising software agents to groups with respect to the thresholds that defines the group membership, only trustworthy experiences are considered. Moreover, in this framework group composition provides a computationally lighter and more expressive means to filter unfair or divergent experiences. The proposed approach does not consider the abusing agents as "misbehaving" agents that may unconditionally be excluded, but merely as agents with diverging view on appreciation. This is a central issue as the setting does not support division between misbehaviour and correct behaviour; merely between trustworthy and untrustworthy.

As a level of trustworthiness relies on experiences, a follow up critic is the distribution of an experience among subjects and proposition. This is not an issue when inter-agent trustworthiness is given. However, when this is not the case, this paper proposes an even distribution to the directly dependent subjects. Even distribution is obviously only possible given that a single experience need not to be dogmatic, i.e. not additive. This underlines the importance of the three-valued metric that also provides the possibility of expressing initial ignorance. With initial ignorance, the groups are initially empty and no vacuous subjects are ever introduced to the groups. Related work, for example [33], faces this same issue and solves it by assigning a priori trusted agents and/or a certain probability to selecting a vacuous agent for transaction in order to broader the domain of knowledge.

Privacy issues with respect to revealing intimate information have been acknowledged by irreversible abstraction, i.e. decay, abstracted experiences and groups. Hence, the framework provides a sense of privacy. Elaborating on this is possible by introducing a predicate defining the software agents to whom abstracted experiences are revealed. If this predicate defines a cardinality of groups, providing experiences only to such whose cardinality exceeds some threshold provides increased privacy. This obviously requires trust to be placed on the group to preserve this.

7. Conclusions

In an ever changing ubiquitous computing environment populated by autonomous agents, no information may be considered unconditionally correct. Hence, the information is subjective. To master the subjectivity, experience based probabilistic methods may be applied. An opinion that base on experiences builds up from initial ignorance. To capture this, Subjective Logic that base on Belief functions is applied.

Having the Subjective Logic as a mathematical framework, this paper proposes a novel view on how such uncertain probabilities may be used to form groups that bond by likes. A group is defined as a set of software agents. Therefore, the experiences a group provides on a subject in proposition is its members' experiences. We provide the means to derive this. Moreover, this paper proposes how groups may be customised by a logical operator. Consequently, it is possible for a software agent to ascertain its opinion on a subject in a proposition by a set of filtered referrals.

8. References

[1] T. Strang, C. Linnhoff-Popien, “A Context Modeling Survey”, in Workshop on Advanced Context Modelling, Reasoning and Management, 2004.

[2] M. Weiser, “The Computer of the Twenty-First Century”, Scientific American, September 1991.

[3] G. Banavar, A Bernstein, “Software infrastructure and design challenges for ubiquitous computing applications”, Commun. ACM, vol. 45, no. 12, pp. 92-96, 2002.

[4] M. Baldauf, S. Dustdar, F. Rosenberg, “A survey on context-aware systems”, International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, no. 4, pp. 263 -277, 2007.

[5] C. Bettini et al., “A survey of context modelling and reasoning techniques”, Pervasive Mob. Comput., vol. 6, no. 2, pp. 161-180, 2010.

[6] A. Dey, Providing Architectural Support for Context-Aware Applications, Georgia Institute of Technology, PhD Thesis 2000.

[7] P. Dockhorn Costa, Architectural support for context-aware applications: from context models to services platforms, Centre for Telematics and Information Technology, University of Twente, PhD thesis, 2007.

[8] R. Schmohl, U. Baumgarten, “Context-aware Computing: a Survey Preparing a Generalized Approach”, in Proc. of the Int. MultiConference of Engineers and Computer Scientists, 2008.

[9] A. Jøsang, “Artificial Reasoning with Subjective Logic”, in Second Australian Workshop on Commonsense Reasoning, 1997.

[10] A. Jøsang, “Trust-Based Decision Making for Electronic Transactions”, In Proceedings of the 4th Nordic Workshop on Secure Computer Systems (NORDSEC’99), 1999.

[11] A. Jøsang, “A logic for uncertain probabilities”, Int. J. Uncertain. Fuzziness Knowl.-Based Syst., vol. 9, no. 3, pp. 279-311, 2001.

[12] A. Jøsang, “Subjective Logic”, Draft book Available at: http://persons.unik.no/josang/papers/subjective_logic.pdf , visited 01.03.2012, Unpublished.

[13] J. Dunn, “The Concept of Trust in the Politics of John Locke”, Cambridge University Press, Cambridge, 1984, vol. Philosophy in History.

[14] C. Castelfranchi, R. Falcone, “Principles of Trust for MAS: Cognitive Anatomy, Social Importance, and Quantification”, In Proc. of the 3rd Int. Conf. on Multi Agent Systems, 1998.

[15] R. Falcone, C. Castelfranchi, “Social trust: a cognitive approach”, In Trust and deception in virtual societies, Kluwer Academic Publishers, pp 55-90, 2001.

[16] P. Bonatti, C. Duma, D. Olemdilla, N. Shahmehri, “An Integration of Reputation-based and Policy-based Trust Management”, In Proc. Semantic Web and Policy Workshop, 2005.

[17] J. Golbeck, Computing with social trust, J. Golbeck, Ed.: Springer, 2009.

[18] T. Grandison, Trust Management for Internet Applications, Imperial College London, PhD Thesis 2003.

[19] M. Blaze, J. Feigenbaum, J. Lacy, “Decentralized Trust Management”, In Proceedings of the 1996 IEEE Symposium on Security and Privacy, 1996.

[20] M. Carbone, M. Nielsen, V. Sassone, “A Formal Model for Trust in Dynamic Networks”, BRICS Report Series Publications. RS-03-4, 2003.

[21] D. Gambetta, “Can We Trust Trust?” in Trust: Making and Breaking Cooperative Relations.: Department of Sociology, University of Oxford, chapter 13, pp. 213-237, 2000.

[22] A. Jøsang, R. Ismail, C. Boyd, “A survey of trust and reputation systems for online service provision”, Decis. Support Syst., vol. 43, no. 2, pp. 618-644, 2007.

[23] H. McKnight, N. Chervaney, “The Meanings of Trust”, Technical Report Working Paper Series 96-04 1996.

[24] B. Christianson, W. Harbison. Why Isn't Trust Transitive?. In Proceedings of the International Workshop on Security Protocols, T. pp. 171-176. 1996.

[25] A. Jøsang, R. Hayward, S. Pope, “Trust network analysis with subjective logic”, In Proceedings of the 29th Australasian Computer Science Conference, vol. 48, pp. 85-94, 2006.

[26] P. Massa, P. Avesani, “Trust Metrics on Controversial Users: Balancing Between Tyranny of the Majority and Echo Chambers”, Int. Journal on Semantic Web and INformation Systems, vol. 3, no. 1, 2007.

[27] A. Abdul-Rahman, S. Hailes, “Supporting Trust in Virtual Communities”, In Proceedings of the 33rd Hawaii International Conference on System Sciences, 2000.
[28] J. Schneider, G. Kortuem, J. Jager, S. Fickas, Z. Segall, “Disseminating Trust Information in Wearable Communities”, Personal Ubiquitous Computing, vol. 4, no. 4, pp. 245-248, Jan. 2000.
[29] J. Sabater, C. Sierra, “Social ReGreT, a reputation model based on social relations”, SIGecom Exch., vol. 3, no. 1, pp. 44-56, Dec. 2001.
[30] S. Buchegger, J-Y. Le Boudec, “A Robust Reputation System for Peer-to-Peer and Mobile Ad-hoc Networks”, in P2PEcon 2004, 2004.
[31] L. Mui, M. Mohtashemi, A Halberstadt, “A Computational Model of Trust and Reputation for E-businesses”, In Proceedings of the 35th Annual Hawaii international Conference on System Sciences Hicss, 2002.
[32] A. Jøsang, R. Ismail, “The beta reputation system”, In Proceedings from the 15th Bled Conference on Electronic Commerce, 2002.
[33] S. Kamvar, M. Schlosser, H. Garcia-Molina, “The Eigentrust algorithm for reputation management in P2P networks”, In Proceedings of the 12th international Conference on World Wide Web, pp. 640-651, 2003.
[34] L. Page, S. Brin, R. Motwani, T. Winograd, “The PageRank Citation Ranking: Bringing Order to the Web”, Technical Report, Stanford InfoLab. 1999.
[35] K. Krukow, Towards a theory of trust for the global ubiquitous computer, University of Aarhus, PhD Thesis, 2006.
[36] A. Whitby, A. Indulska, J. Josang, “Filtering out unfair ratings in bayesian reputation systems”, In Proceedings of the Third International Joint Conference on Autonomous Agenst and Multi Agent System, pp. 106-117, 2004.
[37] A. Jøsang, D. McAnally, “Multiplication and Comultiplication of Beliefs”, International Journal of Approximate Reasoning , vol. 38, no. 1, pp. pp.19-51, 2004.
[38] P. Flocchini, F. Luccio, “Routing in Series Parallel Networks”, Theory of Computing Systems, vol. 2, no. 36, pp. 137-157, 2003.
[39] A. Jøsang, T. Bhuiyan. Optimal Trust Network Analysis with Subjective Logic. Proceedings of the Second International Conference on Emerging Security Information, Systems and Technologies (SECURWARE 2008), Cap Esterel, France, August 2008.
[40] A. Jøsang, S. Pope, M. Daniel, “Conditional Deduction Under Uncertainty”, In Proceedings of the 8th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, pp. 824-835, 2005.
[41] S. Pope, A. Jøsang, “Analysis of Competing Hypothesis using Subjective Logic”, In Proceedings of the 10th International Command and Control Research Technology Symposium, 2005.
[42] A. Jøsang, “Conditional Reasoning with Subjective Logic”, Journal of Multiple-Valued Logic and Soft Computing, vol. 1, no. 15, pp. 5-38, 2008.
[43] J.-R. Abrial, Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010.
[44] A. Jøsang, S. Pope, S. Marsh, “Exploring Different Types of Trust Propagation”, In Proceedings of the 4th International Conference on Trust Management (iTrust'06), pp. 179-192, 2006.
[45] Y. Tang, H. Wang, W. Dou, “Trust based incentive in P2P network”, IEEE International Conference on E-Commerce Technology for Dynamic E-Business, pp.302-305, 2004.
[46] C. Dellarocas, “Immunizing Online Reputation Reporting Systems Against Unfair Ratings and Discriminatory Behavior”. In ACM Conference on Electronic Commerce, pp. 150-157, 2000.

Turku Centre for Computer Science
TUCS Dissertations

1. **Marjo Lipponen**, On Primitive Solutions of the Post Correspondence Problem
2. **Timo Käkölä**, Dual Information Systems in Hyperknowledge Organizations
3. **Ville Leppänen**, Studies on the Realization of PRAM
4. **Cunsheng Ding**, Cryptographic Counter Generators
5. **Sami Viitanen**, Some New Global Optimization Algorithms
6. **Tapio Salakoski**, Representative Classification of Protein Structures
7. **Thomas Långbacka**, An Interactive Environment Supporting the Development of Formally Correct Programs
8. **Thomas Finne**, A Decision Support System for Improving Information Security
9. **Valeria Mihalache**, Cooperation, Communication, Control. Investigations on Grammar Systems.
10. **Marina Waldén**, Formal Reasoning About Distributed Algorithms
11. **Tero Laihonen**, Estimates on the Covering Radius When the Dual Distance is Known
12. **Lucian Ilie**, Decision Problems on Orders of Words
13. **Jukkapekka Hekanaho**, An Evolutionary Approach to Concept Learning
14. **Jouni Järvinen**, Knowledge Representation and Rough Sets
15. **Tomi Pasanen**, In-Place Algorithms for Sorting Problems
16. **Mika Johnsson**, Operational and Tactical Level Optimization in Printed Circuit Board Assembly
17. **Mats Aspnäs**, Multiprocessor Architecture and Programming: The Hathi-2 System
18. **Anna Mikhajlova**, Ensuring Correctness of Object and Component Systems
19. **Vesa Torvinen**, Construction and Evaluation of the Labour Game Method
20. **Jorma Boberg**, Cluster Analysis. A Mathematical Approach with Applications to Protein Structures
21. **Leonid Mikhajlov**, Software Reuse Mechanisms and Techniques: Safety Versus Flexibility
22. **Timo Kaukoranta**, Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization
23. **Gábor Magyar**, On Solution Approaches for Some Industrially Motivated Combinatorial Optimization Problems
24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming – C++ is more Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-Disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Market Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Frank Tétard**, Managers, Fragmentation of Working Time, and Information Systems

41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations Between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity – A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-Type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images
63. **Tommi Meskanen,** On the NTRU Cryptosystem
64. **Saeed Salehi**, Varieties of Tree Languages
65. **Jukka Arvo**, Efficient Algorithms for Hardware-Accelerated Shadow Computation
66. **Mika Hirvikorpi**, On the Tactical Level Production Planning in Flexible Manufacturing Systems
67. **Adrian Costea**, Computational Intelligence Methods for Quantitative Data Mining
68. **Cristina Seceleanu**, A Methodology for Constructing Correct Reactive Systems
69. **Luigia Petre**, Modeling with Action Systems
70. **Lu Yan**, Systematic Design of Ubiquitous Systems
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory

86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming

128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations
132. **Riikka Vuokko**, A Practice Perspective on Organizational Implementation of Information Technology
133. **Jeanette Heidenberg**, Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches
134. **Yong Liu**, Solving the Puzzle of Mobile Learning Adoption
135. **Stina Ojala**, Towards an Integrative Information Society: Studies on Individuality in Speech and Sign
136. **Matteo Brunelli**, Some Advances in Mathematical Models for Preference Relations
137. **Ville Junnila**, On Identifying and Locating-Dominating Codes
138. **Andrzej Mizera**, Methods for Construction and Analysis of Computational Models in Systems Biology. Applications to the Modelling of the Heat Shock Response and the Self-Assembly of Intermediate Filaments.
139. **Csaba Ráduly-Baka**, Algorithmic Solutions for Combinatorial Problems in Resource Management of Manufacturing Environments
140. **Jari Kyngäs**, Solving Challenging Real-World Scheduling Problems
141. **Arho Suominen**, Notes on Emerging Technologies
142. **József Mezei**, A Quantitative View on Fuzzy Numbers
143. **Marta Olszewska**, On the Impact of Rigorous Approaches on the Quality of Development
144. **Antti Airola**, Kernel-Based Ranking: Methods for Learning and Performace Estimation
145. **Aleksi Saarela**, Word Equations and Related Topics: Independence, Decidability and Characterizations
146. **Lasse Bergroth**, Kahden merkkijonon pisimmän yhteisen alijonon ongelma ja sen ratkaiseminen
147. **Thomas Canhao Xu**, Hardware/Software Co-Design for Multicore Architectures
148. **Tuomas Mäkilä**, Software Development Process Modeling – Developers Perspective to Contemporary Modeling Techniques
149. **Shahrokh Nikou**, Opening the Black-Box of IT Artifacts: Looking into Mobile Service Characteristics and Individual Perception
150. **Alessandro Buoni**, Fraud Detection in the Banking Sector: A Multi-Agent Approach
151. **Mats Neovius**, Trustworthy Context Dependency in Ubiquitous Systems

Turku Centre *for* Computer Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www. tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Division for Natural Sciences and Technology
- Department of Information Technologies

ISBN 978-952-12-2808-7
ISSN 1239-1883

Mats Neovius

Trustworthy Context Dependency in Ubiquitous Systems

Mats Neovius

Trustworthy Context Dependency in Ubiquitous Systems