KNOWLEDGE INTEGRATION FOR PROBLEM SOLVING IN THE DEVELOPMENT OF COMPLEX AEROSPACE SYSTEMS

by

MARC GEORGE HADDAD

B. Aerospace Engineering, Georgia Institute of Technology, 1992
M.S. Aerospace Engineering, Georgia Institute of Technology, 1993
M.S. Transportation Engineering, Georgia Institute of Technology, 1998

SUBMITTED TO THE ENGINEERING SYSTEMS DIVISION
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
May 2008
[June 2008]

Signature of Author: _____
Engineering Systems Division
May 28, 2008

Certified by: _____
Deborah J. Nightingale (Thesis Committee Chair)
Professor of the Practice of Aeronautics and Astronautics and Engineering Systems

Certified by: _____
Kirkor Bozdogan (Thesis Supervisor)
Principal Research Associate, Center for Technology, Policy, and Industrial Development

Certified by: _____
Daniel E. Hastings (Thesis Committee Member)
Professor of Aeronautics and Astronautics and Engineering Systems
Dean of Undergraduate Education

Accepted by: _____
Richard Larson (Education Committee Chair)
Mitsui Professor of Engineering Systems and Civil and Environmental Engineering
Chairman, ESD Education Committee

This page intentionally left blank.

KNOWLEDGE INTEGRATION FOR PROBLEM SOLVING IN THE DEVELOPMENT OF COMPLEX AEROSPACE SYSTEMS

by

Marc George Haddad

Submitted to the Engineering Systems Division
on May 2, 2008
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Engineering Systems
Technology, Management and Policy Track

ABSTRACT

The development of complex products requires widespread knowledge interactions among a significant number of individuals and teams designing numerous interrelated components. Increasing product complexity typically leads to a corresponding increase in the types and sources of knowledge that need to be tapped during development, and a common strategy for managing product complexity is to outsource parts and components to external suppliers. As a result, the knowledge required for development is increasingly specialized and distributed across multiple boundaries spanning large-scale organizational networks, thus requiring the subsequent integration of this knowledge in order to accomplish the development task. A framework for knowledge integration in the development of complex systems in a large-scale organizational context is proposed in this thesis using an extensive review of the pertinent literature. The framework consists of the main channels, strategies, practices and mechanisms most commonly used to transfer, share and apply knowledge in the course of complex technical problem solving. The framework is progressively refined using empirical data collected through several rounds of interviews and a questionnaire instrument administered across three major aircraft programs in the defense aerospace industry. We find that knowledge integration in routine problem solving situations is most efficiently and effectively accomplished through extensive transfer and sharing of codified information using formal mechanisms such as information systems, while knowledge integration for major non-routine troubleshooting events requires extensive integration of individual expertise and know-how through both formal and informal advice sharing as well as direct assistance across internal and external organizational boundaries. A principal contribution of this research is in demonstrating how different characteristics of the engineering artifact defined in terms of product complexity, architecture and technology newness, and different aspects of problem solving including problem type and novelty, drive the knowledge integration process and the organizational system. We conclude that permeability of cross-program boundaries, direct relationships with functional groups and rich tacit knowledge flow from suppliers are critical for countering rampant firefighting in complex product development.

Thesis Supervisor: Kirkor Bozdogan
Title: Principal Research Associate, Center for Technology, Policy, and Industrial Development

"True glory consists in doing what deserves to be written;
in writing what deserves to be read."

- Gaius Plinius Secundus (Pliny the Elder)

I hope this work will be useful to academics and practitioners alike.

EXECUTIVE SUMMARY

Knowledge integration is an emerging discipline in organizational science and the related knowledge-based body of literature where the principal idea is that the increasing scale of organizational arrangements and the increasing complexity of products under development combine to disperse knowledge resources across multiple boundaries spanning large-scale networks, and as a result there is a need for organizations to continuously gather their knowledge resources in order to maintain their ability to innovate and to sustain their competitive position in the market. This process also enables the organization's problem solving capabilities since knowledge is synonymous with problem solving, especially in complex product development where the required knowledge is increasingly specialized, varied (multi-disciplinary) and distributed across external boundaries with partners and suppliers. Knowledge integration in this context is done through a process of transferring knowledge from multiple sources in the organizational network to the locus of problem solving (which are typically the product teams tasked with development), combining that knowledge at the recipient site and using it in the course of problem solving. It follows that large-scale organizations engaged in complex product development are best served to have the appropriate conduits, policies and devices for transferring, sharing and applying knowledge to solve problems.

The primary purpose of this research is therefore to identify the main characteristics for knowledge integration in large-scale complex product development. For research lens, the choice of military avionics is adopted due to their high level of complexity and the richness of the defense context in terms of major barriers facing the knowledge integration process. To that end, four main research questions are posited, specifically: (1) What are the types and sources of engineering knowledge in the development of complex avionics systems?, (2) What are the strategies, practices, channels and mechanisms for integrating knowledge to solve technical problems in this context? (3) How is knowledge integration informed by the organizational environment and the characteristics of the problem at hand?, and (4) What are the technology, management and policy issues in this context? A framework for knowledge integration is proposed in this thesis using an extensive review of the pertinent literature on knowledge integration, problem solving, organization design and complex product development. The

framework is then progressively refined using empirical data collected through 70 interviews with 50 individuals and 49 problem solving cases collected through a questionnaire instrument administered across three major military aircraft programs in the defense aerospace industry. The grounded theory method is used to analyze the interview data in order to develop the main concepts for knowledge integration, and a comparative analysis of the questionnaire data is used to determine the relationships between knowledge integration and different product-specific and problem-specific characteristics, namely product complexity, system architecture and technology newness, as well as problem type and problem novelty, respectively. Several heuristics are proposed to that end.

A main conclusion from this research is that knowledge integration is markedly different for routine versus non-routine complex problem solving situations. We find that knowledge integration for routine problem solving is most efficiently and effectively accomplished through extensive transfer and sharing of codified information through formal structures and mechanisms, while knowledge integration for major non-routine troubleshooting events requires extensive integration of individual expertise and know-how through formal and informal advice sharing and direct assistance across internal and external organizational boundaries. Of particular importance is the sharing of lessons learned from past problem solving events, which was found to be critical for dealing with complexity but ineffective when relying on information technology-based mechanisms for the integration process. The research also leads to the finding that modular product development typically leads to problem solving in isolation, while integrated product architectures compromise rapid troubleshooting capability. An original contribution from this research is therefore in demonstrating how the engineering artifact drives the organizational system through the knowledge integration process. We also find that the integration of knowledge across program boundaries is severely hampered by the disconnected relationships between programs in large-scale project-based organizations which are typical in complex product development environments. Similarly, indirect relationships with functional groups and arm's length relationships with suppliers were found to hamper the effectiveness of knowledge integration. We conclude that permeability of cross-program boundaries, supplemented organizational forms and team structures, and rich tacit knowledge flow with suppliers are critical for countering rampant firefighting in complex product development.

This page intentionally left blank.

ACKNOWLEDGEMENTS

To my dear Samar, for the infinite love you give,

To my parents, Grace and George, for all you have sacrificed for my sake,

To my brothers, Tony and Elie, for putting up with me,

And to my best-friends and extended family, for the inspirations, and for the dream…

I wish to extend my thanks first and foremost to all three members of my doctoral committee: to Professor Deborah Nightingale, the committee chair, for your encouragement and always helpful interventions, in true Lean spirit you always managed to support me with "the right thing at the right time", but most of all for your uplifting kindness; to my advisor and research supervisor and mentor Dr. Kirk Bozdogan, for the long hours you gave in hands-on support of this work, the brainstorming and the insights, the motivation and the tremendous patience throughout, it is an understatement to say that this thesis would not have been possible without your intellectual guidance and help; and to Dean Daniel Hastings whose classes I greatly enjoyed, whose advice was always greatly valuable, and whose academic support got me through my first steps at MIT.

I wish to acknowledge the Lean Aerospace Initiative (LAI), the organization that supported this research both financially and with the best and most ample of resources, and the lab community that made life on campus not only easier but enjoyable, even in sub-zero weather. Thank you to Dr. Eric Rebentisch, Dr. Ricardo Valerdi and Mr. Tom Shields for all your assistance, and to all my LAI colleagues and friends, some of you officers and all of you gentlemen and women, thank you for the stimulating conversations, with special thanks to Alexis for all your generous help.

Last but certainly not least, I wish to thank all the companies and individuals who participated in this research for the valuable time and invaluable insights you gave, I cannot cite you by name as I'm bound to protect company confidentiality, but I wish to especially recognize Lee W., Jim W. and Scott S. for scheduling and coordinating my numerous site visits and teleconference calls, thank you for making this research happen and for making it enjoyable as well.

This page intentionally left blank.

TABLE OF CONTENTS

This page intentionally left blank.

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

AESA	Active Electronically Steered Array
AI	Artificial Intelligence
AMC	Advanced Mission Computer
BFE	Buyer Furnished Equipment
CDR	Critical Design Review
CIP	Common Integrated Processor
CNI	Communications, Navigation and Identification
COMINT	Communications Intelligence
COTS	Commercial-Off-The-Shelf
CPR	Common Problem Report
DOD	Department of Defense
ECCM	Electronic Counter-Counter Measures
ECM	Electronic Counter Measures
ELINT	Electronic Intelligence
ESM	Electronic Support Measures
EW	Electronic Warfare
EO	Electro Optics
GHz	Gigahertz
GPS	Global Positioning System
ICP	Integrated Core Processor
IFF	Identification Friend-Or-Foe
IMA	Integrated Modular Avionics
ILS	Instrument Landing System
IPT	Integrated Product Team
IT	Information Technology
IR	Infrared
IRAD	Independent Research and Development
ITAR	International Traffic in Arms Regulations
JSF	Joint Strike Fighter

MASA	Modular Avionics System Architecture
MC	Mission Computer
MF Radar	Multi-Function Radio Aid to Detection and Ranging
MHz	Megahertz
MMC	Modular Mission Computer
MWS	Missile Warning System
NASA	National Aeronautics and Space Administration
TACAN	Tactical Air Navigation
TWS	Track-While-Scan
OSA	Open System Architecture
PDR	Preliminary Design Review
RF	Radio Frequency
RWR	Radar Warning Receiver
SA	Synthetic Aperture
SATCOM	Satellite Communications
SDD	System Design and Development
SIGINT	Signals Intelligence
SME	Subject Matter Expert
SRR	Systems Readiness Review
T/R	Transmit/Receive
UHF	Ultrahigh Frequency
USAF	United States Air Force
VHF	Very High Frequency
WBS	Work Breakdown Structure

Greek Characters

χ^2	Chi-square goodness-of-fit test statistic
α	Statistical significance level

1. INTRODUCTION

The development of complex products requires widespread knowledge interactions between a significant number of individuals and teams designing numerous interrelated component parts (Eppinger 2002). Increasing product complexity typically leads to a further increase in the types and sources of knowledge that need to be tapped during development, and a common strategy for managing design complexity is to outsource component parts of the product to external suppliers (Baldwin and Clark 1997). As a result, the knowledge required for development becomes increasingly distributed across multiple boundaries spanning large-scale organizational networks, thus requiring the subsequent integration of this knowledge in order to accomplish the development task. The process of integrating knowledge to that end involves, in the first place, the *transfer* of knowledge in the form of codified information and varied expertise, know-how and skills from different sources with varied organizational affiliations (typically including the customer, partners and/or suppliers, as well as other programs and functions in the prime organization). However, since knowledge transferred is not necessarily absorbed, integration also involves *sharing* the transferred knowledge at the recipient site.

It is also well established that "no matter how good the systems engineering process, it can only succeed by the *application* of the skills and experience of individuals and teams, and successful interactions between multidisciplinary organizations" (Moir and Seabridge 2006). This means that the highest value-adding activity in product development comes during the actual use of the knowledge acquired from multiple sources and disciplines in the course of problem solving. In large-scale development of complex products, problem-solving occurs mostly at the level of the integrated product teams (IPT's) responsible for different parts of the product and engaged at different levels of the product design hierarchy (Browning 1996). Therefore, knowledge integration, as defined in this thesis, consists of transferring knowledge in the form of individual expertise and codified information from multiple teams across the organizational network, combining it with existing knowledge through a process of knowledge sharing at the recipient site, and applying the newly created knowledge to solve a specific problem. It is thus a highly contextual process, and the mechanisms employed vary considerably depending on the nature of the problem, the characteristics of the product system in question, and the relationships of the

stakeholders involved, among other factors. This thesis will identify the main characteristics of the knowledge integration process in complex product development environments in terms of the most important channels, strategies, practices, and mechanisms for transferring, sharing and applying knowledge in different problem solving situations.

1.1 Problem Statement

The nature of complex product development is such that no single team or organizational entity has all the knowledge resources needed to tackle the entire development task on its own. It is frequently the case that complex problem solving requires the collaboration of multiple stakeholders with separate individual interests and sometimes having conflicting goals. The main question of interest in this research is therefore about how large-scale organizations are able to collaboratively integrate their knowledge resources in order to develop highly complex products, and to what extent are common collaborative arrangements effective in this context.

1.2 Motivation

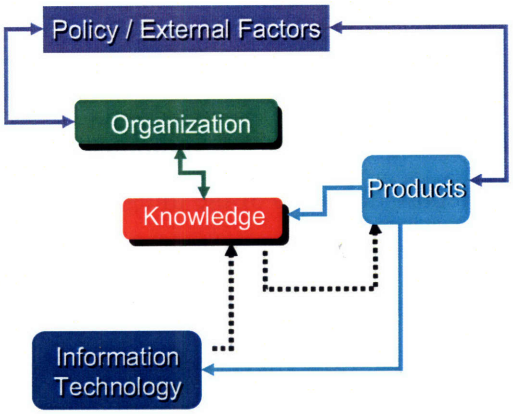
The need for integrating diverse knowledge from multiple sources across large-scale organizational boundaries is a real world problem for organizations engaged in the development of complex systems, as already outlined in the introduction of this thesis. Furthermore, the knowledge integration phenomenon is currently poorly understood both in the literature and in practice, especially in terms of the lack of empirical evidence about how organizations can better integrate knowledge in order to solve complex problems more efficiently and effectively (De Boer, Van Den Bosch et al. 1999; Hansen, Nohria et al. 1999). This has direct consequences on the time and cost of developing complex products since some of the major sources of cost and schedule overruns in complex product development are the time and resources employed to troubleshoot unforeseen technical problems at various stages of the development process. Therefore, this constitutes the primary motivation for this research which aims to provide an improved understanding of the knowledge integration phenomenon, as well as to provide recommendations for how organizations can better integrate their knowledge resources and sustain their competitive advantage.

The second main motivation in this research is the need for a deeper and more holistic exploration of the knowledge integration phenomenon in different problem solving contexts, and specifically in how the use of different knowledge integration conduits and devices can benefit the expensive and difficult troubleshooting of major technical problems. Troubleshooting in engineering environments is accomplished by applying the organization's problem-solving capabilities in order to diagnose and solve problems (Fujimoto 1999), with problem solving becoming increasingly complicated in line with the growing difficulty of the problems encountered. In complex product development, problem difficulty is a function of the complex interdependencies at the level of the product design itself, as well as at the level of the stakeholder relationships between the numerous teams and organizations involved in the problem-solving effort (Braha and Bar-Yam 2007), which makes it imperative for the developing organization to streamline its problem solving process in order to accomplish complex tasks efficiently and effectively. Organizations employ several routine and non-routine conduits and devices to that end, all of which involve the integration of knowledge from different sources inside and outside the organization (Grant 1996a). This makes the integration of knowledge a key enabler of problem solving and a major factor in the bottom line of any complex product development project. Therefore, the second motivation for this research is the lack of a holistic examination and analysis of the relationships between the knowledge integration process and the troubleshooting environment and the need to determine how knowledge can be integrated more efficiently and effectively in different troubleshooting contexts.

This investigation is especially relevant for *large-scale* product development where knowledge resources are typically spread thin over multiple programs, as well as being often poorly allocated relative to the needs of each program or project to begin with, which amplifies troubleshooting events into a continuous, chaotic and wasteful phenomenon known as "firefighting" (Repenning 2001). As a result, large-scale organizations engaged in complex product development have a vested interest in evolving their technical problem-solving capabilities through more efficient and effective integration of their engineering knowledge resources, thus reducing the cost and time to develop complex systems (Takeishi 2002). Therefore, this research carries direct benefits to large-scale organizations involved in complex

product development in terms of providing new and practical insights for how to better leverage their dispersed knowledge resources and enhance their competitive position in the market.

A final motivation of this research is in the need to address the theoretical gaps in the literature on knowledge integration, namely the lack of clear and precise definitions of the most essential elements and processes that form the basis of the integration phenomenon. The body of literature on knowledge integration being an emerging and relatively nascent thrust in the knowledge view of organizational science, it is rife with high-level, vague and often conflicting definitions and interpretations, which presents an opportunity for theoretical contributions in terms of improved definitions at both the conceptual and operational levels. This not only contributes to the field of knowledge integration itself, but also carries implications for related theory such as organizational and inter-organizational learning where the focus is on how organizations can share and absorb new knowledge across internal and external boundaries in ways that do not compromise their competitive advantage. An improved understanding of the principles and strategies for integrating knowledge will therefore provide new insights for learning across organizational boundaries since knowledge integration is by definition based on mutual learning through collaboration during problem solving. Figure 1 situates this work in the overall study of organizations and product development, with knowledge being at the core of all organizational processes, and the integration of knowledge being enabled by (but not restricted to) information technology and having implications for organizational and product evolution.



Adapted from: Nightingale and Rhodes, 2006

Figure 1: Situating the Research

1.3 Research Objectives

I begin this work on the topics of knowledge integration and complex problem solving with an expressive quote from the television sitcom "Frasier", where TV psychiatrist Frasier Crane helps people with their problems through his radio show, and where the words to one of the jingles in the show say: "If you've got a problem, if you're feeling low, looking for some answers, things you need to know, all you've got to do is ask, all you've got to do is ask" – *"Frasier", Season 7, "They're playing our song"*. Even as I had heard this jingle several times in the past, I only came to appreciate the importance of the simple and elegant message within as I engaged in this work, where it became apparent that half of the battle to solving any problem, no matter how simple or difficult, is in knowing where to get the right knowledge, and how to get it.

The primary objective in this thesis is to identify the most commonly used channels, strategies, practices and mechanisms for integrating knowledge across team and organizational boundaries in a complex product development context, focusing on military avionics development as a research lens as it offers a rich context for examining a wide variety of complex problems and policy challenges facing the knowledge integration process.. The primary outcome from this research is therefore in the development and validation of a conceptual framework describing the knowledge integration process in military avionics development, with generalizable conclusions for similar technology-based complex engineering environments in other industries.

The second main objective of this research is to empirically validate existing theory about knowledge integration in different complex product development environments by performing a comprehensive analysis of how specific elements of the integration process are used in different problem solving contexts with varying levels of complexity. Specifically, the research will frame the knowledge integration process for different types of development problems and different product architectures, as well as different levels of problem novelty and technology maturity. The outcome from this part of the investigation will provide a better understanding of the knowledge integration phenomenon and allow for practical recommendations for how

organizations can better integrate their knowledge resources in specific problem solving environments.

The third objective of this research is to provide general heuristics for knowledge integration in terms of rules of thumb for how organizations can better enable the knowledge integration process under different problem solving conditions with varying degrees of system and organizational complexity. Secondary objectives in this research are 1) bolstering the theoretical foundations for the emerging field of knowledge integration with conceptual and operational definitions currently absent or poorly defined in the existing literature, and; 2) identifying the most important enablers and barriers most commonly encountered in complex development environments, and providing recommendations in terms of best practices for organizational integration and complex problem solving.

In summary, and compared with previous studies which only focused on the transfer or sharing part of knowledge integration, such as the investigation of the frequency of inter-team communication in complex product development (Sosa, Eppinger et al. 2000b), the main contribution from this research is in a) providing a more complete framing of the knowledge integration phenomenon (including the transfer, sharing and application of knowledge in complex problem solving), and b) in framing knowledge integration under different troubleshooting contexts and offering recommendations for improvement.

1.4 Research Questions

Following from the discussion in the previous section on the primary objectives in this research, the main research question can be articulated as follows:

How do large-scale organizations integrate diverse knowledge from multiple sources across internal and external boundaries in order to solve complex problems efficiently and effectively during the product development?

Four secondary questions frame the investigation of this research, as follows:

1. What are the types and sources of engineering knowledge in the development of complex systems?

2. What are the channels, strategies, practices and mechanisms for integrating knowledge in this context?

3. How is knowledge integration informed by characteristics of the problem and the organizational context at hand?

4. What are the technology, management and policy issues in this context?

In the process of addressing these questions, a theoretical framework for knowledge integration will be developed in this thesis and validated empirically in the field.

1.5 Thesis Outline

This thesis is organized along the same sequence of steps followed in the course of developing and validating the conceptual framework for knowledge integration, as detailed below. In addition to the main chapters, four appendices are also provided that detail the research protocol adopted in this work and summarize major steps and outcomes from the data analysis.

Chapter 2 is a review of the three main bodies of literature applicable in this work, namely the literature on knowledge integration and problem solving, complex product development and organizational design. The chapter summarizes and connects together the main insights of most relevance for this research. In addition, conceptual and operational definitions for knowledge integration are developed in this chapter and grounded in existing theory.

A preliminary knowledge integration framework developed based on existing insights is then proposed in Chapter 3, along with typologies for knowledge types, channels and integrative

mechanisms which will serve to guide the field inquiry into the knowledge integration phenomenon.

Chapter 4 gives an overview of military avionics systems as the research lens in this work. A description of the different systems of interest, their main roles and functions, and the evolution of system complexity and architecture as the main characteristics of interest in this research are also given.

Chapter 5 provides an overview of the research cases investigated in the field and the research methods and procedures used to collect the data, namely the grounded theory method for interview data collection and analysis, and a questionnaire instrument for collecting frequency data about different knowledge integration characteristics in specific problem solving situations.

Chapter 6 deals with data analysis for both qualitative and quantitative data, and offers a series of refinements for the originally proposed knowledge integration framework based on the data analysis outcomes.

Finally, Chapter 7 presents major conclusions, implications and recommendations from this research, and summarizes the main theoretical and practical contributions as well as potential future directions for expanding on this work.

2. LITERATURE REVIEW

The academic void that this research aims to address is in the absence of a theoretical framework for how large-scale enterprises integrate their knowledge resources within and across organizational boundaries in order to solve technical problems in the development of complex product systems. This puts the investigation in this work at the intersection of the main bodies of literature on knowledge management, organization theory and complex product design and development, namely:

a. *Knowledge Integration theory,* specifically the literature addressing the types and characteristics of knowledge in organizations (Nonaka 1994; Von Hippel 1994) and the integration channels, mechanisms and strategies employed for integrating knowledge in organizations and large-scale networks (Grant 1996b; Okhuysen and Eisenhardt 2002). The concept of knowledge integration is the basis for the emerging knowledge-based view of the firm (Kogut and Zander 1992; Grant 1996a; Nickerson and Zenger 2004) which concerns itself with the efficient appropriation of knowledge for production, and while the theory behind it remains relatively undeveloped, it draws on and parallels significant previous work in such areas as knowledge transfer (Aoshima 2002; Carlile 2004), knowledge sharing (Dyer and Nobeoka 2000; Hansen 2002), organizational learning (Senge 1994; Argote 1999) and boundary spanning (Star 1989; Carlile 2002).

b. *Problem Solving theory,* specifically the literature addressing how individuals and teams search for and acquire knowledge from across the organization in order to solve technical problems in complex product development (Clark and Fujimoto 1991; Hansen 2002; Cross and Sproull 2004). This literature provides useful insights for understanding the key characteristics of problems and problem solving approaches that are typical in this context (Jonassen 2000; Postrel 2002), as well as for determining how problem solving is enabled by different knowledge integration channels, mechanisms and strategies in different problem situations (Daft and Lengel 1986; Eisenhardt and Tabrizi 1995; Nickerson and Zenger 2004).

c. The literature on *Organization Design,* specifically the areas addressing the influence of organizational and team structure on problem solving (Robertson and Langlois 1995; Dosi, Hobday et al. 2000) and on information sharing and transfer (Browning 1996; Christensen, Verlinden et al. 1999). This literature provides insights into which knowledge integration strategies/practices, channels and mechanisms are appropriate in different organizational contexts.

d. The literature on *Complex Product Design and Development*, specifically those contributions highlighting the knowledge interactions and actors involved in the system design and integration processes, and the influence of key product attributes such as system architecture, complexity and technology maturity on coordinating interactions at the team and organizational levels (Henderson and Clark 1990; Sako 2003). Inferences from this literature are used for establishing a link between the product development and the knowledge integration processes (Sanchez and Mahoney 1996; Hoopes and Postrel 1999; Sosa and Eppinger 2002).

The area where this research is expected to make a contribution can be visualized in Figure 2 below:

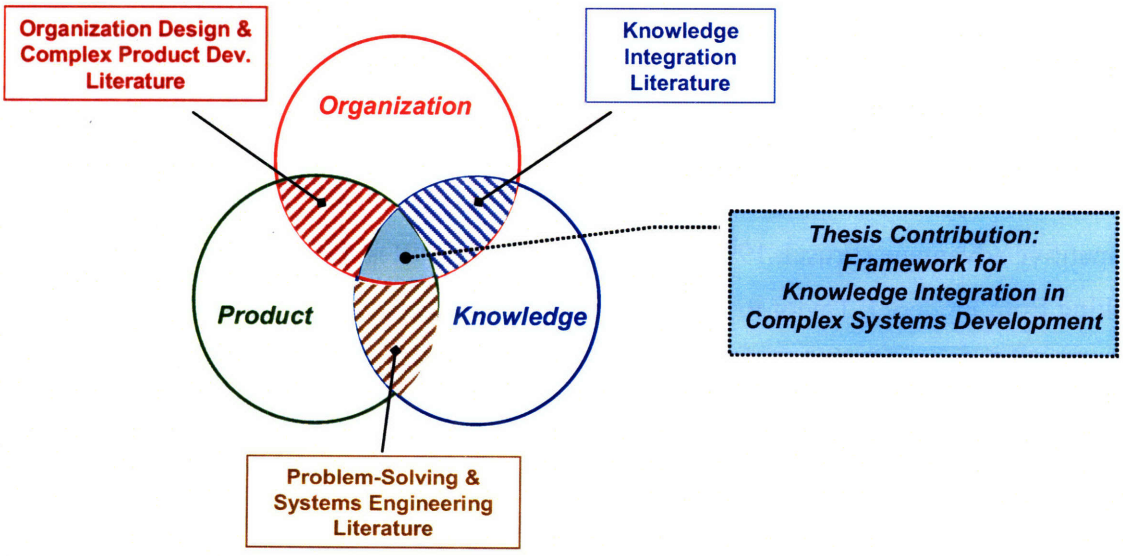


Figure 2: Thesis Contribution Area

Following is a review and summary of the major insights from previous work in the three areas introduced above.

2.1 Insights from the Literature on Knowledge Integration

All contemporary perspectives in organization theory seem to converge on the relatively recent realization that knowledge is the most strategic resource of the firm (Nonaka 1994; Grant and Baden-Fuller 1995; Conner and Prahalad 1996), that it is central to all production activities (Prahalad and Hamel 1990), and essential for creating and sustaining competitive advantage (March 1991; Nonaka and Takeuchi 1995; Prusak 1996). This is because knowledge (comprised of information, know-how, technology and skill) has become the differentiator between firms, especially in the context of developing high-technology complex products and services where rapid technological evolution and product/process innovation can make or break even the most established of firms (Utterback 1996; Grant 1996a).

Several of these emerging perspectives have been concerned with developing a new knowledge-based theory of the firm (Demsetz 1988; Kogut and Zander 1992; Foss 1993; Spender 1996; Grant 1996a), with many of those contributions reflecting an evolution of the earlier resource-based view of the firm where knowledge now replaces physical assets such as capital and infrastructure to become the most valuable and useful resource in the organization (Conner and Prahalad 1996). The knowledge-based view further advocates that the organization's primary role is to leverage its knowledge resources efficiently and effectively for production and competitive advantage (Prusak 1996; Nonaka and Teece 2001). This is in contrast with previous thinking from Frederick Taylor to Herbert Simon that placed emphasis solely on the information part of knowledge, such as that captured in rules, procedures and work practices and where the organization was mainly seen as an information-processing entity concerned with efficiency in coordinating and managing information in order to increase output, independently of the actors involved and of the larger environment in which the organization exists (Simon 1973). The knowledge-based view instead recognizes that knowledge is information supplemented with context and experience, in other words knowledge that is most important for organizations is not

just the data or scientific formulas or principles that can be expressed in words and numbers and which can be readily found in reports and manuals, but it is also the interpreted and enacted form of this information held by experienced and specialized members of the organization (Nonaka and Takeuchi 1995). The organization's primary concern then becomes one of *integrating* (sometimes the term "combining" is used) all of its dispersed knowledge resources in order to apply them in production, and as a means of creating new knowledge out of different and novel combinations of existing knowledge (Grant 1996a). It is precisely this process by which organizations integrate knowledge that is the concern of this thesis, where the focus is on understanding the "how-to" of knowledge integration in a particular organizational context. But before addressing what constitutes knowledge integration, the following section will start by providing an overview of what is meant by the term "knowledge" in the organizational context, and how the term is used in this thesis in particular.

2.1.1 Defining "Knowledge" in Knowledge Integration

Much has been written in economics and organization theory about the related concepts of knowledge and information, starting with the pioneering views of the economist Alfred Marshall who was the first to advocate that "*knowledge is the most powerful engine of production*", and who advanced a positivist view of knowledge as an objective and fixed asset to be efficiently utilized by the organization. Later works by Friedrich Von Hayek and Michael Polanyi defined the powerful concept of tacit (or implicit) knowledge held by individuals, also known as subjective knowledge, and that is evolved through personal experience (Polanyi 1966), with the organization being a repository of this knowledge and its structure a means for maximizing knowledge utilization. These early views of knowledge (i.e. the objective and subjective views) gave two important but very different perspectives with little convergence between them, and in fact it can be argued that the two are diametrically opposed since the subjective view of knowledge resonates more with the anti-positivist philosophy. However, the positivist view of knowledge remained dominant in the literature through much of the early period of the information age, where knowledge was seen as a tangible asset belonging to the organization instead of the individual (Penrose 1955), and which was paralleled in practice with a consistent focus on maximizing the efficiency of knowledge exploitation by the organization. The

objective view of knowledge was also later reinforced by Herbert Simon's framing of the organization and its members as information processors, a concept rooted in his earlier principle of "bounded rationality" describing the cognitive limitation on the ability of individuals to process information (Simon 1973). What this meant for the organization was that the design of authority and decision-making structures had to minimize information overload on the individual; as a result, and in order to make up for this human limitation, organizations had to increase their collective capacity for processing information through machines and infrastructure (Simon 1973; Galbraith 1974). The Simonian principle was later put into wide practice with the modern revolution in information technology when organizations turned their attention almost exclusively to the implementation of information systems for moving and managing codified information quickly and cheaply. But the overemphasis on information processing in organizations over the past few decades came at the expense of developing and retaining the more dynamic and valuable tacit knowledge of individuals, and it wasn't until recent success stories from the Japanese tradition in knowledge management (as demonstrated by the success of the Japanese manufacturing industry) that attention in professional practice turned again to the tacit dimension of knowledge (Womack, Jones et al. 1991; Nonaka and Takeuchi 1995; Womack and Jones 1996).

The ensuing reaction against the information-centric view of knowledge generated a host of perspectives still common today that completely separate knowledge from information and data. In this reactionary view, the use of the term "knowledge" is short for describing personal tacit knowledge exclusively, namely to the exclusion of objective information and data. The practical driver behind this differentiation was the recent realization of the significant value of individual tacit knowledge in production, something that was previously ignored by organizations in favor of superior information processing capabilities. The theoretical underpinning of this view is the reasoning that knowledge is exclusive to the individuals who create it and develop it out of their own personal experiences, making it a very personal asset (Nonaka 1994). Knowledge is thus considered synonymous with the subjective beliefs and values of each individual (or group of individuals), and as such cannot be confused or lumped with objective facts and observations. In that sense, knowledge and personal knowledge become one and the same, while objective

information and data are seen as separate from personal knowledge and held separately in books and organizational repositories.

While the above appears to be a largely semantic differentiation, it is nonetheless important to note it here for the purposes of this work since it bears direct consequence on what constitutes knowledge in the organization, and therefore what ultimately constitutes knowledge integration which is the central concern of this thesis. Specifically, the limited focus on tacit knowledge under the previous definition downplays the importance of information and data in production, and ignores their role in various knowledge processes in the organization. This not only contradicts reality in everyday practice, but is also incompatible with widely accepted theory about knowledge creation and learning in organizations where knowledge is considered both tacit and explicit (Nonaka and Takeuchi 1995). In this modern view, both subjective beliefs and objective information are seen as essential components of knowledge[1] which complement each other in a closed-loop cycle of knowledge creation, such that explicit knowledge is the result of articulating tacit knowledge and codifying it into generic information, whereas tacit knowledge is evolved by internalizing information to learn and develop new skills. Similarly in recent perspectives on knowledge integration, knowledge is described as inclusive of information, technology, know-how and skills (Grant 1996a). Here information refers to the codified part of knowledge that is already captured in documents or electronic format; know-how refers to the tacit or subjective knowledge of individuals that is developed through experience and which can be embodied or embedded in technologies, products and tools; and skill refers to the innate personal knowledge of individuals evolved through practice and learning-by-doing. This last definition does not separate information from knowledge, but it does not lump or confuse the two concepts together either.

Given the overview of the theory presented above, the practical question for this research as posed at the outset in section § 2.1 remains: *What is knowledge in knowledge integration?* The historical debate clearly suggests that the inclusive knowledge definition is more useful from the

[1] (Machlup, 1978) defines information as a constituent part of knowledge while making a distinction between the two concepts – information is described as a flow of messages which might add to, restructure or change knowledge that is anchored on the commitment and beliefs of its holders. (Nonaka, 1994) elaborates on this definition to describe the knowledge creation process as the interplay between tacit knowledge and explicit information.

perspective of knowledge integration as it does not exclude the objective part of knowledge from the integration process, while at the same time keeping a distinction with the subjective form of knowledge. Therefore, there is more value for this work in adopting the more inclusive view since it leads to a more comprehensive framework of the knowledge integration process. This is in contrast to the exclusivist view which would lead to the exclusion of information and data from the investigation.

Specifically, in this thesis I use the term knowledge to refer to both the objective (raw data and information) held by the organization, and the subjective (know-how and skills) held by individuals, and I adopt the established view that different types of knowledge are not created equal in that there is a knowledge hierarchy where the subjective knowledge of individuals is the most valuable for competitive advantage. This hierarchy is illustrated in the "knowledge pyramid" shown in Figure 3 below.

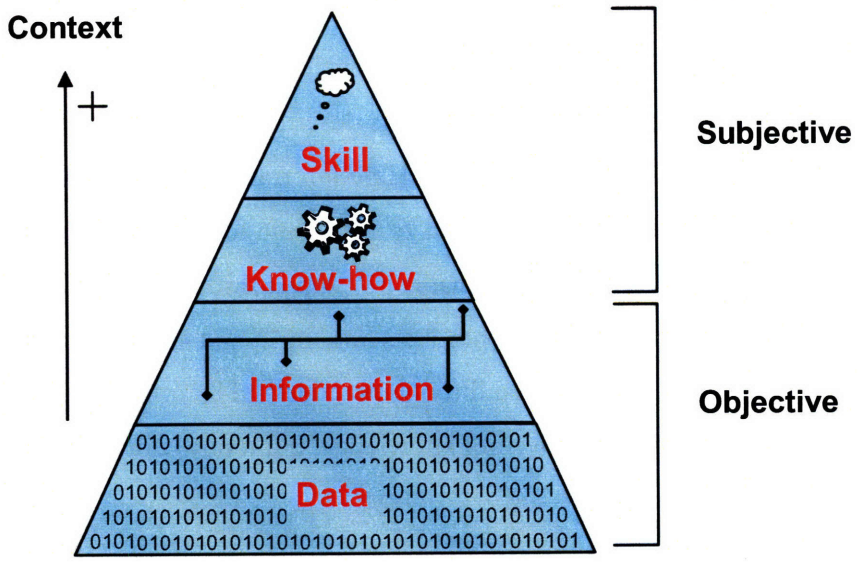


Figure 3: The Knowledge Pyramid

As the above figure illustrates, the hierarchy in the composition of knowledge is based on subjective context, with personal knowledge (or tacit knowledge) being the most contextual and specialized, while impersonal knowledge (or explicit knowledge) is generic and more abstract. Each level of the pyramid builds on the previous level through added context, with raw data

(such as the output of a testing process) as the least contextual form of knowledge, followed by information which is made up of raw data that has been processed and put into some context[2] (such as charts and tables that establish relationships between the data), yielding structured or unstructured observations and facts; know-how is then information supplemented with analysis and interpretation or deduction (and can be embodied in physical technologies or products); and finally skill is know-how supplemented with further experience and innate abilities. The process of contextualizing generic data into highly specialized skill is illustrated in Figure 4.

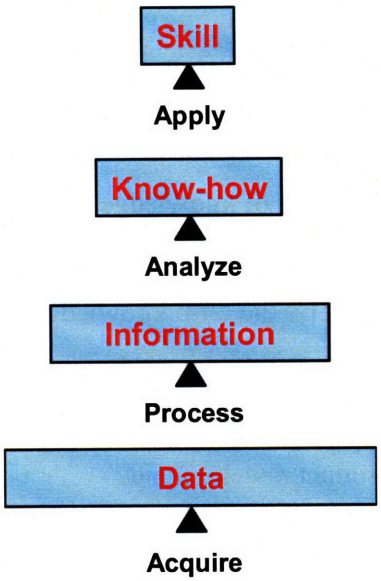


Figure 4: The Build-up of Knowledge

The above figure shows that while knowledge includes information and data, they do not by themselves constitute knowledge in its modern definition. It also shows that individual knowledge takes the most processing and is therefore hardest to build, thus it provides the organization having access to it with competitive advantage. Furthermore, individual knowledge is indeed a higher level of knowledge as shown in Figure 4 since it is the type of knowledge that requires the cognitive processing of individuals. However, it is important to note that in an

[2] Note that explicit knowledge is not only created from the bottom up by adding context to raw data and information as defined here, but also from the top down by abstracting and capturing tacit know-how and personal skills, such as the explicit knowledge found in science or mathematics books which has been generalized to be true in any context. In both cases, explicit and objective knowledge are considered synonymous since they are both abstract and generic.

organizational context, all types of knowledge have their usefulness in solving problems and accomplishing tasks, and are valued differently in different situations.

In summary, this section has introduced the two main types of knowledge that are important for production in organizations, the first is the subjective knowledge of individuals (also referred to as tacit, implicit, or personal knowledge) which takes the form of individual know-how and skills, and the second is objective knowledge (also referred to as explicit, codified or impersonal knowledge) made up of data and information and which is captured in written or electronic format. This thesis is concerned with both subjective and objective types of knowledge, and specifically with *how* they are integrated in a large-scale complex product development context. The concept of knowledge integration will be defined in the following section. Subsequent sections will further address the specifics of knowledge and knowledge integration in the context of complex product development. That is, this section has only addressed the question: "What is knowledge?" to clarify what is meant by the term knowledge in its most general sense; further discussion in § 2.1.6 on component and architectural knowledge will address the question: "Knowledge about what?" to specify what types of knowledge are of interest in this thesis within the specific context of complex product development.

2.1.2 An Operational Definition of Knowledge Integration

As already discussed in the previous two sections, the knowledge-based view in the literature advances the argument that knowledge most relevant for production is created and held by individuals in the course of performing tasks as well as during socialization and reflection (Nonaka and Takeuchi 1995). This means that the organization's knowledge resources are distributed and dispersed across the organization, especially in large-scale multi-program enterprise networks where much of the production knowledge resides in the supplier base, outside of traditional firm boundaries (Baldwin and Clark 1997). Therefore organizational performance in the knowledge era is no longer dependent on coordinating tasks and managing information only, but rather on the ability of the focal firm to continuously integrate its dispersed "pockets" of specialized knowledge efficiently and effectively (i.e. in novel and sustainable ways) in order to carry out its production activities and maintain competitive advantage (Kogut

and Zander 1992; Purvis, Sambamurthy et al. 2001) However despite the wide consensus in the literature on the prominence and centrality of knowledge in production activities and the role of the organization as a knowledge integrator, there is still very little theory on what constitutes knowledge integration (Brown and Duguid 2001), and even less on how this integration is accomplished in practice in terms of the actual organizational channels and mechanisms for integrating knowledge (De Boer, Van Den Bosch et al. 1999; Takeishi 2002). Indeed the concept of knowledge integration remains fairly conceptual, meaning that it is still at a fairly high-level of aggregation and lacking a sufficiently detailed common operational definition. Thus a prerequisite to researching the mechanics of knowledge integration is to start with a definition that is more specific and better scoped than what is provided in the current literature.

A starting point for a conceptual-operational definition is the one proposed by (Grant 1996a) in which knowledge integration is "a process for coordinating the specialized knowledge of individuals". While this definition clarifies what is involved in knowledge integration (e.g. a coordination process involving the tacit knowledge of individuals), it remains tautological from an operational perspective, in the sense that it describes integration as coordination, which is an equally open-ended concept by Grant's own disclaimer that "*organization theory lacks a rigorous ... well developed and widely agreed theory of coordination*", or as Herbert Simon describes it in more extreme terms: "... *'coordination' is what we say when we don't know what we are talking about*" [3]. Grant goes on to define integration in terms of broad categories of mechanisms for coordinating knowledge between individuals depending on the interdependency of the task they need to accomplish between them, which can be summarized as 1) communication systems, documents and routine procedures for coordinating explicit information, and 2) group problem-solving for coordinating the personal know-how and experience of individuals. Grant's definition of integration is thus very much in line with contemporary views of organizational coordination as "managing dependencies between activities" using "processes of information transfer and group decision making" (Malone and Crowston 1991; Malone and Crowston 1994), so much so in fact that it makes it even harder to distinguish between what constitutes integration versus coordination.

[3] Even in more recent attempts at developing a theory of coordination, the authors (Malone and Crowston 1991; Malone and Crowston 1994) acknowledge the difficulty inherent in defining the concept of organizational coordination in operational terms, and provide a long list of diverse definitions commonly used in the literature.

In addition, the difficulty with Grant's approach is that it makes the operational part of integration (in other words the mechanisms of knowledge integration) as the basis for defining the concept itself rather than the other way around, where a clearly defined concept is the basis for how to operationalize it. In fairness to Grant, who is widely considered as the father of this concept and one of the pioneers of the knowledge-based view of the firm, his approach did draw clear boundaries as to what knowledge integration is not. However, the lack of clarity at the inception and early definition of the concept left the door open for many speculative and equally open-ended or even conflicting definitions that followed. It is thus that many authors have come to define knowledge integration as a collection of many related and unrelated activities from knowledge creation to acquisition, transfer, storage, utilization and even maintenance of knowledge, and sometimes all at once – see for example (Yang 2005). At the other extreme, some definitions are minimalist to the point of being at an even higher level than the starting definition, where the concept of integration is simply reformulated as "absorbing" and "blending" – see for example (Balaji and Ahuja 2005). In between these two extremes are a host of definitions that confuse integration with coordination, communication, cooperation, and/or collaboration, among others. However this variety in defining the concept of knowledge integration is not all due to the lack of a clear foundational definition; it is in fact indicative of another difficulty inherent in the concept of integration itself, namely that it is indeed a multi-faceted process involving activities which are overlapping and often cannot be clearly separated[4].

But when considering the dispersed nature of knowledge in the organization, especially in large-scale organizational networks where knowledge is distributed across vast boundaries, knowledge integration becomes first and foremost synonymous with acquiring and assembling knowledge from diverse sources in the course of practice, first acquiring knowledge to where it is needed through established relationships between source and recipient, and then assembling (or combining) it with the receiver's current knowledge so that the resultant knowledge can be used to accomplish a task. Integration is therefore accomplished when the organization is able to perform a task that it could not complete with existing knowledge alone. In this sense, the

[4] Merriam Webster dictionary defines integration as the process of "uniting with something else" or "incorporating into a larger unit".

resultant (integrated) knowledge is greater than the mere combination of acquired and existing knowledge since new (additional) knowledge may be created in the combination process, as well as in the process of putting the combined knowledge to practical use. Therefore, integration by this definition is distinct from and superior to acquisition and combination alone despite any colloquial similarity in terms; the latter are in fact subsets of integration and clearly not equivalent to the complete process of integration. Thus, staying true to Grant's original intent for what constitutes knowledge integration, I offer the following conceptual definition:

> Conceptual Definition for Knowledge Integration:
> Knowledge integration is *bringing* diverse knowledge from multiple sources *to bear* on a complex problem or task.

The highlighted terms above embody the key ideas in the concept of knowledge integration as already posited in the existing literature. And while the term "bringing knowledge to bear" is largely conceptual, it can in fact be made operational in a number of ways that do not leave room for conflicting interpretations since it can be mapped to distinct organizational processes, namely the acquisition of new knowledge from diverse sources through established relationships between source and recipient, its combination with existing knowledge at the recipient site and the utilization of the resultant knowledge in the course of practice. In organizational terms, acquiring knowledge translates to a process of *transfer* between source and recipient, combining knowledge translates to a process of *sharing* with different members or groups at the recipient site, and using the resultant knowledge in practice translates to a process of *applying* it to accomplish a task, which is typically in the course of *problem-solving*. In other words, the knowledge integration process consists of sub-processes that involve the transfer, sharing and application of knowledge in order to solve problems. I note here that a common claim in the literature is that knowledge transferred and/or shared is not necessarily appropriated or absorbed by the receiver, often due to the absence of a common knowledge base or due to syntactic or semantic differences (Cohen and Levinthal 1990; Carlile 2002). It follows that knowledge appropriation is an important aspect of the knowledge integration process, and it can be argued in principle that the integration process cannot be fully described without taking the appropriation part into consideration. However, as noted previously in this section, the appropriation or

absorption concept is very difficult to operationalize explicitly, such that it is impossible to talk about channels or mechanisms for knowledge absorption for example. Instead, the literature on organizational boundaries (sometimes also referred to as knowledge boundaries) incorporates the underlying factors affecting knowledge absorption, namely the differences in syntax and semantics noted above, into different types of boundaries that knowledge must be integrated across (Star 1989; Carlile 2004). As such, I include the concept of organizational boundaries as one of the main dimensions in my definition of knowledge integration, and I assume that knowledge which has been successfully transferred across boundaries and successfully applied to solve problems has been necessarily absorbed to some degree already (section § 2.1.9 addresses in more detail the underlying factors related to knowledge absorption). Expanding on the conceptual definition offered above, I propose the following operational definition:

> Operational Definition for Knowledge Integration:
>
> Knowledge integration is the process of *transferring* knowledge, both tacit and explicit, across organizational boundaries, *sharing* it with individuals and teams at the recipient site, and *applying* the resultant knowledge to solve problems.

This definition builds on the original concept proposed in (Grant 1996a) while bounding the process of integration to clear and unambiguous sub-processes used in practice. An example of tacit knowledge integration by this definition is when multiple individuals are transferred from different organizations to form a taskforce where they share and apply knowledge together in order to accomplish a complex task. Similarly, an example of explicit knowledge integration is when previous solution information, such as a solution template, is transferred from a database and customized to solve a complex problem. Integration is therefore accomplished only when knowledge is transferred, shared and applied. As a result, integration must be inclusive of all three sub-processes and cannot be considered equivalent to knowledge transfer alone or knowledge sharing by itself for example. However, in some cases it is not necessary to carry out one or more sub-processes in order to achieve integration, such as for example when the knowledge required is already on site (e.g. resident on the team), and thus does not need to be transferred across boundaries.

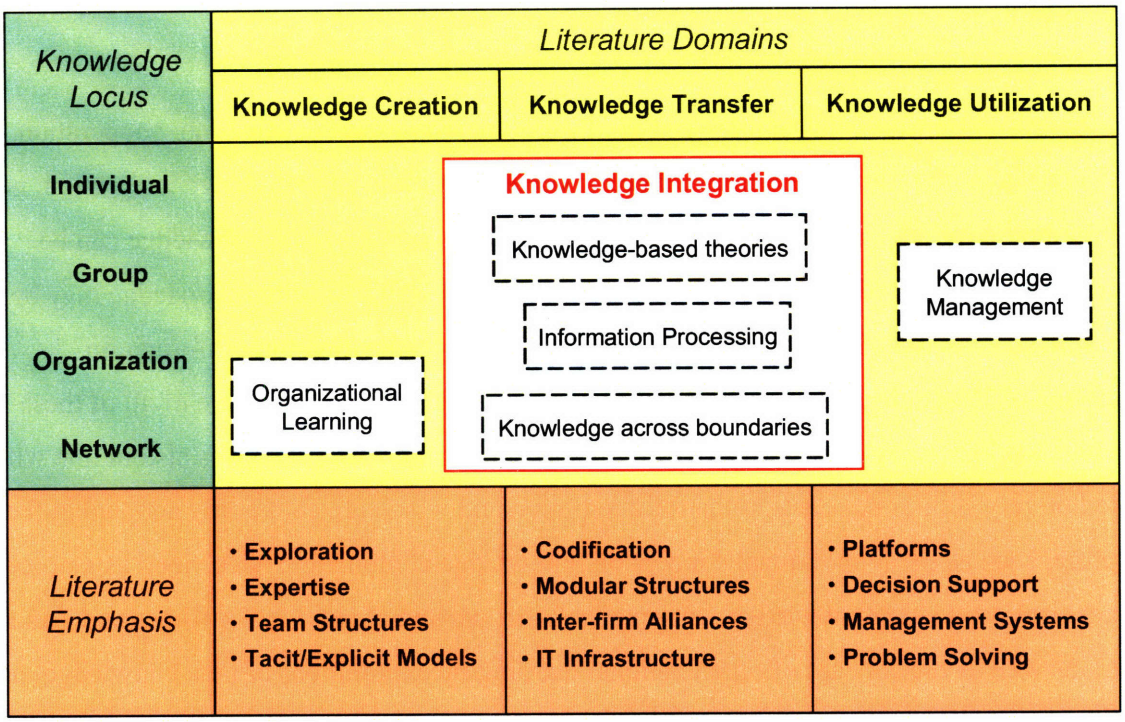
The definition is scoped to include only those sub-processes which are most relevant to the concept of integration (encompassing the "where", "what" and "how" of knowledge integration) and that are readily observable in practice (in contrast to such abstract concepts as absorbing or blending), which makes it an operational definition for the purposes of this research. It is important to note here that while this definition makes the concept of knowledge integration operational, it does not define the concept itself in terms of any operational specifics as is the case in the definition proposed by Grant. In other words, the actual mechanisms for transferring, sharing and applying knowledge do not form the basis for defining what integration means, this is why they do not figure in the definition proposed here. Instead, the concept is defined independently of any of the means for "how-to" operationalize it.

Also, by this definition, integration is not identical to processes of coordination, cooperation, collaboration or communication as advanced in other definitions of knowledge integration, nor is it contradictory or exclusive of these processes (e.g. sharing or applying knowledge can be accomplished by individuals collaborating on a task, just like transferring knowledge may be accomplished by communication systems or individuals coordinating their information resources). In that sense, knowledge integration by this definition is inclusive of all of these activities without being confused with or limited to one or more of them. It is also in line with related or very similar concepts to that of integration and which are commonly adopted in the literature, such as the foundational concept of "knowledge combination" described as a process of "acquiring and using knowledge...in practice" (Brown and Duguid 1991; Kogut and Zander 1996), as well as the concept of organizational learning by exploration (of new knowledge) and exploitation (of existing knowledge) (March 1991).

In addition, the proposed definition is consistent with the widely established view that the value of knowledge is fully realized when it is interacted in a closed-loop cycle of socialization, combination, internalization and externalization known as the SECI framework for knowledge creation (Nonaka and Takeuchi 1995) since the first two (socialization and combination) are processes where knowledge is transferred and shared, while the latter (internalization and externalization) are processes by which individuals apply knowledge and learn-by-doing. Consistency between the two concepts (i.e. integration and creation) is important since new

knowledge is created out of the integration process and therefore the two concepts are not entirely distinct or orthogonal. Finally, this definition is complete and purposeful in terms of highlighting the role of the knowledge integration process as an enabler of problem solving (Carlile 2002; Nickerson and Zenger 2004), and specifically in a team environment as originally proposed in (Grant 1996a).

Figure 5 below situates the concept of knowledge integration (as defined above) in the overall literature on knowledge.



Adapted from Choo & Bontis, 2002

Figure 5: Knowledge Integration in the Literature

As Figure 5 illustrates, knowledge integration in the existing literature concerns itself mostly with the process of transferring / sharing knowledge across organizational boundaries, while also being connected to the utilization (or application) part of the overall literature on knowledge, and to a lesser extent with the literature on knowledge creation (since new knowledge is created out of the integration process). The operational definition for knowledge integration provided in this

section is in line with the general principles advanced in these bodies of literature in terms of the fundamentals for "how" knowledge is created, transferred and utilized in an organizational context.

2.1.3 The Gap in the Literature on Knowledge Integration

With an operational definition of knowledge integration in hand as specified in the previous section, I propose to address in this thesis the academic void about *how* knowledge is integrated in organizations, specifically the strategies/practices[5], channels and mechanisms[6] by which knowledge is transferred, shared and applied in large-scale organizational networks in order to solve problems in the development of complex products. As already highlighted by several authors, there is a gap in the literature on framing the mechanics of the knowledge integration process in practice. According to (De Boer, Van Den Bosch et al. 1999), while knowledge integration has been explored in some detail as a concept, there is a lack of insights about what firms actually do to integrate their knowledge: *"...the use of the term combination of* (Kogut and Zander 1992) *runs parallel to the term integration used by* (Grant 1996a), *and the term configuration used by* (Henderson and Clark 1990). *What is neglected in most publications, however, is a specification of the different combination or integration mechanisms a firm has at its disposal..."* Similarly, (Hansen 2002) argues that the knowledge transfer literature *"does not shed much light on the integrative mechanisms that would allow one business unit to obtain knowledge from another."*

As discussed in the previous section, there are some first-order insights in the literature on how knowledge is integrated in organizations, such as about the use of systems, documents and procedures as mechanisms for integrating explicit knowledge, versus group problem solving for integrating the skills and know-how of individuals (Grant 1996a). However, these insights remain very general and there is currently no research on the mechanics of knowledge integration in specific contexts, such as determining the exact integration mechanisms that are

[5] The terms "strategies" and "practices" are used interchangeably in this thesis to refer to broad action plans set by an organization with a primary or ultimate objective of integrating knowledge (e.g. a strategy of knowledge reuse or a practice of sharing lessons learned).
[6] The term "mechanism" is used to refer to the specific means by which knowledge is integrated under a particular strategy/practice (e.g. a database of lessons learned).

most frequently used in a particular organizational context, or which group structure is most efficient and effective in a particular problem solving context. The current literature does not address the mechanics of knowledge integration in any great detail either, such as determining the specific channels along which certain mechanisms are most typically used, or which specific types of systems, documents and procedures are used to integrate knowledge along a particular channel.

The absence of insights about the mechanics of knowledge integration is especially relevant for research in complex product development where the organizational context typically consists of large-scale networks of highly dispersed knowledge "pockets", and where the development process requires the integration of very diverse knowledge resources. In such situations, knowledge interactions can involve multiple parties and can be affected by several environmental factors, from the structure of the developing organization to the characteristics of the product under development, among others, with different product characteristics and organizational forms requiring different types of channels, mechanisms, strategies and practices for integrating the required knowledge to accomplish the development task. (Takeishi 2002) makes this case in his discussion of knowledge integration and management across prime-supplier boundaries when he says *"...evidence on organizational mechanisms for effective knowledge management remains somewhat anecdotal... We need to build more empirical, wide-ranging studies of products with different architectures."*

Based on the evidence above about the lack of insights on the mechanics of the knowledge integration process, the remainder of this chapter will be focused on exploring the existing literature with an eye for the "how-to" of knowledge search, transfer, sharing and application during problem solving, especially in a large-scale complex development context. I begin by reviewing a sample of seminal studies which have already explored different characteristics and factors affecting knowledge integration, as shown in Table 1 below. This table presents an overview of the most cited and most recent literature relevant to the operational aspects of knowledge integration, illustrating the major insights from various perspectives on the topic.

Table 1: Overview of the Literature on Knowledge Integration

Reference	Focus	Method	Relevant Conclusions/Results
Ancona & Caldwell, 1992	External team interactions with the environment	Conceptual & Hypothesis-test	Vertical negotiation and horizontal task coordination as well as scouting for technical knowledge increase team performance
Aoshima, 2002	Knowledge transfer across product generations	Hypothesis-test	Transfer system knowledge by rotating engineers; transfer component knowledge by documents and information systems
Carlile, 2004	Knowledge integration across syntactic, semantic and pragmatic boundaries	Conceptual & Empirical	IT systems, liaison individuals and negotiators or modelers to transfer, translate and transform knowledge respectively
De Boer et.al, 1999	Knowledge integration as a function of organizational forms and capabilities	Conceptual & Case-study	Integrating design and architectural knowledge relies on the firm's socialization, coordination and information systems capabilities
Dyer & Nobeoka, 2000	Knowledge integration through collective learning routines across organizations	Conceptual & Empirical	Network-wide communities, people rotation, dedicated resources and free assistance to members increase network learning
Edmondson & Sole, 2002	Knowledge integration to bridge gaps across geographically dispersed IPT's	Conceptual & Case-study	IPT members compensate for knowledge gaps by drawing on broader and deeper expertise and skills in communities of practice
Grant, 1996 (a)	Knowledge integration as the basis for the knowledge-based theory of the firm	Conceptual	Efficient knowledge integration by using multiple informal and formal mechanisms, tacit and explicit, flexibly and simultaneously
Grant, 1996 (b)	Knowledge integration as the means for evolving organizational capability	Conceptual	Tacit knowledge is integrated by routine tasks and activities, explicit knowledge by codified directives, procedures, technology
Hansen, 2002	Knowledge integration across team boundaries in an organizational network	Conceptual & Hypothesis-test	Direct inter-team connections are beneficial for transferring tacit knowledge, but inefficient for transferring codified knowledge
Hoopes & Postrel, 1999	Product development performance as a function of intra-firm knowledge integration	Conceptual & Case-study	Increasing product complexity requires increased knowledge sharing across boundaries and early specs development
Nonaka & Takeuchi, 1995	Creating new knowledge through a cycle of articulating, sharing, combining, absorbing	Conceptual	Knowledge must spiral up from individuals to groups and across organizational boundaries in order to realize its value
Okhuysen & Eisenhardt, 2002	Formal interventions for improving group flexibility and knowledge integration	Hypothesis-test	Questioning others enables knowledge integration in groups, while information sharing internally has little to no impact
Szulanski, 1996	Impediments to knowledge transfer inside the organization	Conceptual & Empirical	Knowledge ambiguity, lack of trust and arm's length relationships impede the transfer of knowledge inside the organization

The preceding overview of the literature in Table 1 above is meant to highlight the main themes and dimensions of relevance to the "how-to" of knowledge-integration, and which will be enumerated and elaborated on with specific insights in the following subsections.

2.1.4 The Tacit / Explicit Dimension

As already introduced in § 2.1.1, one of the most useful and commonly discussed themes in the literature on knowledge in organizations is the distinction between tacit and explicit knowledge (Nonaka and Takeuchi 1995; Spender 1996; Grant 1996a; Hansen, Nohria et al. 1999). In categorizing knowledge based on its many different types, the tacit and explicit characterizations are the two foremost dimensions by which every other kind of knowledge can be further described, from design knowledge to business knowledge and others. (Polanyi 1966) is widely considered to be the authoritative source on the concept of tacit knowledge which he defines as personal knowledge acquired through experience and which is inseparable from an individual's aptitude, beliefs and commitment. In that sense tacit knowing is like riding a bicycle, it is knowledge acquired through experience and becomes an innate skill that we cannot easily describe to others except through personal demonstration. This is why Polanyi argues that tacit knowledge is difficult to transfer to others, something he explains with a famous quote when he says of people: "*we know more than we can tell*" (Polanyi, 1966: p.4). In contrast, explicit knowledge is the part of knowledge that is readily articulated and has been or can be captured in written or electronic format (Nonaka and Takeuchi 1995; Spender 1996), such as low-level information in the form of raw data, or situated information in the form of scientific principles.

To illustrate the fine distinction between tacit and explicit knowledge in practice, consider the recent real-life troubleshooting event in July of 2005 where NASA, a very large-scale organization with massive information resources especially in terms of documented rules and procedures, was unable to troubleshoot a fuel sensor malfunction until the retired engineer who designed part of the system 30 years earlier was brought out of retirement to help diagnose the problem[7]. This is because the knowledge that individuals accrue through long years of experience and specialization, as well as their innate skills at specific tasks, are often difficult to

[7] **http://edition.cnn.com/2005/TECH/space/07/19/space.shuttle/index.html** - accessed May 15, 2007

capture and pass on as codified information. This example shows that tacit knowledge is different from and more valuable than explicit knowledge, without the two being completely independent of each other. To further illustrate this distinction with a more common example, consider the difference between the information found in a cookbook (i.e. the steps in a recipe) and the cooking knowledge of the chef (i.e. the skills and accumulated know-how of the cook). Following a recipe or knowing many recipes does not necessarily make one a great chef; instead it is the innate skills developed over time and the experience from deductions and interpretations through analysis, trial-and-error and learning-by-doing that distinguish a great chef who can create great tasting food from an ordinary cook who can only create ordinary tasting food. Similarly, the distinction between a junior engineer and a senior engineer is not only measured in terms of the amount of information each one retains (indeed a junior engineer may have memorized more information from books and manuals than a senior engineer). However it is the experience of the latter in analyzing and interpreting facts and deducing insights from information that separates his or her skill level from that of a junior engineer (Vincenti 1990).

There is wide agreement in the literature as well as in practice that tacit knowledge is more valuable for organizations than explicit knowledge, since the latter can be easily obtained from books and databases, whereas tacit knowledge is held by individuals who take it with them when they leave the organization (Nonaka and Takeuchi 1995; Grant 1996b). The value of tacit knowledge becomes even more relevant in high technology environments where explicit knowledge becomes obsolete very quickly (Prusak 1996; Davenport and Prusak 1998). Tacit knowledge is also considered as the real source of competitive advantage since it is difficult to imitate (Nonaka and Takeuchi 1995; Prusak 1996; Spender 1996), whereas knowledge that has been codified can be readily absorbed by others (Takeuchi and Nonaka 2004). It is however argued that both tacit and explicit knowledge are interdependent and inseparable (Brown and Duguid 2001), and that they are complementary in terms of their usefulness for production, so that the presence of both is necessary for competitive advantage as they are the two essential components in the process of creating new knowledge (Nonaka and Takeuchi 1995). More simply put, an organization without experienced and highly skilled employees would not be able to compete, while an organization without procedure manuals or computer software would not be able to produce.

When it comes to transferability of knowledge, there are different views in the literature on how much of tacit knowledge is transferable, where most authors argue that at least a part of an individual's tacit knowledge can be transferred to others through observation (such as watching a an artist draw) or learning-by-doing (as in attempting to draw under the artist's supervision), while a minority argue that tacit knowledge is personal skill that is acquired with little help from others and that cannot be taught (Gourlay 2006). In contrast there is wide consensus on the fact that explicit knowledge is easily transferable through documents and information systems (Hansen, Nohria et al. 1999), making it also easy to imitate by competitors.

For the purposes of this research, I adopt the more generally accepted notion that tacit knowledge may be integrated through personal interaction such as face-to-face communication and group interaction, whereas explicit knowledge can be readily integrated through documents and information systems (Grant 1996b). This classification makes up the first dimension in the "how-to" of knowledge integration.

2.1.5 The Formal / Informal Dimension

Knowledge is interacted within and between organizations through formal and informal instruments (i.e. the channels and mechanisms by which knowledge is flowed). Formalized instruments are those instituted by the organization for the purposes of transferring, sharing and applying knowledge, while informal channels are those created and maintained by individual members of the organization (Davenport and Prusak 1998). Both types of instruments are considered of equal importance from the perspective of knowledge integration, where informal channels and mechanisms serve to complement the formalized ones, and as such they both are necessary components of the knowledge integration process (Grant 1996a).

Examples of formalized channels for transferring and sharing knowledge are the coordination links embedded in organizational hierarchies where by virtue of one entity reporting to another, knowledge is formally and routinely flowed between them (Grant 1996a). Formalized channels for transfer and sharing as well as for applying knowledge are also embedded in the

infrastructure of the organization, such as the numerous facilities and systems typically provided for carrying out tasks and activities, from conference rooms to information systems. (Galbraith 1974) identified liaison devices, task forces, and permanent committees as some of the key formal mechanisms for integrating knowledge across multiple teams in an organization. Other examples of formal mechanisms instituted along formal channels are routine or regular meetings, official directives, databases, communities of practice, among countless others. Examples of informal channels are the personal networks and relationships between individuals that are either created and/or maintained by those individuals with little or no formalization by the organization. The mechanisms employed along informal channels are also informal in nature, with some of the more common examples including online communities, after-hours socializing, or even what is known as the "grapevine" in reference to indirect communication channels inside the organization (Johnson, Donohue et al. 1994). Both formal and informal channels can be intra- as well as inter-organizational. However due to the competitive nature of inter-firm relationships, formalized channels are more dominant, such as formalized prime-supplier communication channels through contracts, site visits, and shared databases. Both formal and informal channels have advantages and disadvantages which vary depending on the organizational context and the circumstances governing the knowledge integration process. A good summary of the typical strengths and drawbacks of formal versus informal instruments is given by (Davenport and Prusak 1998): "...the main advantage of informal networks is that they are self-updating and adaptive since they consist of people continuously interacting with each other...In contrast, more formal systems such as electronic repositories become stale as soon as they are established." The above insights are reframed and summarized in Table 2 below:

Table 2: Formal versus Informal Knowledge Integration

Integration Channels	Dominant Mechanisms	Knowledge Integration Characteristics
Prime-Supplier	Formal	Shared systems, site visits, contracts, boundary objects
Intra-firm	Formal	Information systems, teams, meetings, liaison devices
Intra-firm	Informal	Personal networks, online communities of practice

Finally, it is important to note that formal instruments are by definition a means for the organization to formalize knowledge interactions between parties with pre-defined relationships, which reduces ambiguity in the knowledge integration and problem solving processes. As such, formal channels and mechanisms are most efficient when dealing with problems that are highly ambiguous. On the other hand, informal knowledge interactions between friends/trusted colleagues provide a high element of trust for knowledge integration where both parties consider each other as equals and have no apprehension about openly communicating their knowledge. Thus, problems where mediation is necessary can benefit from informal interactions.

2.1.6 The Component / Architectural Dimension

Perhaps the second most commonly discussed dimension (after the tacit/explicit theme) in the modern literature on organizational knowledge is the one pertaining to product development knowledge, namely the distinction between component (or component-specific) knowledge and architectural knowledge (Henderson and Clark 1990; Sanchez and Mahoney 1996; Baldwin and Clark 1997; De Boer, Van Den Bosch et al. 1999; Aoshima 2002; Takeishi 2002). Recalling that a complex product is a design hierarchy comprising several subsystem modules where each is formed by different component technologies (Ulrich 1995), it follows that there are different layers of knowledge corresponding to the product's design structure, including knowledge about component parts, subsystem modules and the overall architecture of the system. Component knowledge, which is also referred to as subsystem knowledge or design knowledge, is then defined as the knowledge an organization or an individual possesses about the product's core design concepts and how they are implemented in particular parts of the system, such as a component part or a subsystem module (Henderson and Clark 1990). In other words, component knowledge is the knowledge required to design and develop one of the "black boxes" in the system, and includes scientific and engineering knowledge (or discipline-specific knowledge) related to the design characteristics and functionality of the box. Architectural knowledge, also known as system or layout knowledge is defined as the knowledge about how the different components or black boxes of a system are integrated together into a coherent whole. More precisely, it is the knowledge about how interdependent components interface together in order

to deliver the required functional performance for the system (Ulrich 1995). Examples of component knowledge of interest in this thesis are knowledge about subsystem requirements, specifications and standards, and knowledge about component materials, functional and structural design, and core technology. Examples of architectural knowledge of interest in this thesis are knowledge about system requirements, specifications and standards, and knowledge about structural and functional coordination between components[8].

The component/architectural distinction is important for knowledge integration since different types of knowledge are more efficiently integrated through different types of channels and mechanisms, as already discussed in § 2.1.4 on the tacit/explicit dimension above. In this case, since component knowledge consists of scientific and engineering knowledge that is readily captured in explicit form, it can be easily transferred, shared and applied through documents and information systems. In contrast, architectural knowledge is considered to be difficult to learn and imitate since it is embedded in the tacit knowledge of the organization (Henderson and Clark 1990). In a recent empirical study, (Aoshima 2002) shows that architectural knowledge is best integrated within an organization by rotating experienced systems engineers from old projects to new ones. In the same study, Aoshima demonstrates that component engineers refer to documents and reports more frequently than system integrators, which validates the view that component design knowledge is more readily integrated through explicit mechanisms, while architectural knowledge integration relies on people interactions.

In complex systems development, suppliers are typically tasked with design and development of parts of the system (i.e. different subsystems that are part of the overall system), while the prime contractor is tasked with architecting the overall system and integrating the various subsystems at the end (Fine and Whitney 1996; Wissmann and Yassine 2005). It is thus considered that most component knowledge is resident in the supplier base, while most architectural knowledge is retained by the prime organization. From this perspective, there is an emphasis both in theory and in practice on the establishment and utilization of explicit knowledge integration channels and mechanisms across prime-supplier boundaries, while tacit integration channels and

[8] Note that these different types of knowledge constitute the "know-what" of knowledge integration in the context of product development and that are of concern in this research specifically, and therefore this knowledge typology provides an answer to the question of "knowledge about what?" as posed in § 2.1.1

mechanisms are more emphasized internally to the one organization (see § 2.3.2 for a further explanation of the influence of product architecture on the nature of knowledge interactions between prime and supplier organizations). This practice is reinforced by the imperative for prime organizations to protect their architectural knowledge as the most important source of their competitive advantage, as well as the need for module suppliers to make the most of their modular innovations by introducing them at the right time and for maximum value (Robertson and Langlois 1995). However, in the context of complex systems development, the increasing demands on component suppliers for greater capability and functionality of components and subsystems they provide, as well as the increasing interdependence between different modules in the overall system designed and assembled by the prime, make it such that suppliers need further visibility into the overall product architecture, while the system integrators need further knowledge of the internal workings of the various components and subsystem modules. This is most evident at the integration stages of complex systems where different supplier-provided components and subsystem modules are assembled together in the overall system, and where troubleshooting of problems necessitates intense knowledge interactions between prime and supplier organizations that go beyond the exchange of explicit component knowledge. In fact, current research findings from the automotive industry show that for complex designs involving new technologies, systems engineers need a high level of component-specific knowledge from the supplier (Takeishi 2002).

I conclude that for complex product development, the integration of component and architectural knowledge cannot be clearly delineated along internal/external organizational boundaries as is typical in the development of relatively simple products. It is therefore imperative to establish both tacit and explicit channels and mechanisms for integrating component and architectural knowledge within and across organizational boundaries, regardless of the role of the organizations involved in the development process.

2.1.7 The Vertical / Horizontal Dimension

(Demsetz 1988) and (Grant 1996a) define organizational boundaries in terms of knowledge dependencies between different stages of production along two dimensions, namely the

horizontal (across different specialties or different projects in the same organization) and vertical (across different organizations or different hierarchies in the same organization). They argue that vertically linked stages of production A and B will be integrated within the same firm if production at stage B requires access to knowledge utilized in stage A, as is the case in the development of tightly interconnected products which are more efficiently developed in-house (efficiency here is equivalent to minimizing high coordination costs across organizational boundaries (Christensen, Verlinden et al. 1999)). In this case, vertical knowledge integration would be internal to the firm, such as between engineering and manufacturing, or between subsystem level and system level teams inside the same program. Otherwise if stage B output can be accomplished independently of stage A, then production can take place in separate firms, as is the case in the development of modular systems where some parts of the system are efficiently outsourced to suppliers as separate modules (Baldwin and Clark 1997). In this case, vertical knowledge integration would be along a channel linking prime and supplier organizations.

Similarly, it is argued that horizontal integration will take place within a single firm in cases of knowledge interdependence between two parallel stages of production, as is the case in multi-product firms (Nobeoka and Cusumano 1994). In this case, horizontal knowledge integration would be between different programs within the same organization. In related lines of research, scholars have also demonstrated the importance of having horizontal linkages between different subunits within the same organization in order to have effective coordination and open knowledge sharing internally. This research has shown that a subunit's information processing capacity is enhanced by horizontal inter-unit integration mechanisms (Galbraith 1974; Hansen, Nohria et al. 1999; Gupta and Govindarajan 2000). However, the current literature is silent on horizontal integration inside a single function or program, such as would be the case between different system teams or between multiple subsystem teams belonging to the same program. This is because of a common assumption in the current literature that integration takes place across traditional organizational boundaries such as those separating the firm and its environment or market (Santos and Eisenhardt 2005), or those separating different organizational entities within the same firm (Clark and Fujimoto 1991), as in integration across different functions (for example, between engineering and manufacturing), across different programs or projects (for

example, integration between different generations of the same product or between independent product lines (Nobeoka 1993)), across different vertical layers in the organizational hierarchy (for example, between management and production), or altogether different and independent organizations (for example, between prime and supplier (Takeishi 2001)). As such, the concept of integration as framed in the literature does not explicitly address the horizontal boundaries between subunits or teams within the same entity and at the same level in the hierarchy, such as system-level or subsystem-level teams belonging to the same program. This is due to the continuing dominance of the static view in the literature which frames the firm in terms of its conventional divisions, thus limiting the extent of horizontal integration to one between traditional or economically separate entities (Foss 1996a). But this does not reflect the reality of knowledge integration in complex product development where large system and subsystem teams often constitute separate autonomous entities even within the same program or function (Browning 1997). In this context, the knowledge integration picture would not be complete without taking into account the horizontal channels linking different teams within the same program or function.

An example of inter-team horizontal knowledge integration is typically seen in the development of hybrid "modular-integral" systems where functional interdependencies between subsystem modules are complex enough that they cannot be fully specified and easily assembled without extensive horizontal interactions between the different subsystem teams. In such cases, which are common in the development of highly complex and customized products such as military aerospace systems (Moir and Seabridge 2006), different teams develop different subsystem modules separately, but are forced to interact together extensively during system integration in order to troubleshoot emergent problems due to the complex interdependencies between the different subsystems. Thus, most of the knowledge integrated at that stage is along the horizontal channels inside the same program at the subsystem team level.

To further complete the characterization of knowledge integration along the vertical and horizontal dimensions, I note the related concept of lateral linkages which is not tackled in the literature on knowledge integration, or which is used synonymously with horizontal

relationships[9] – for example in (Galbraith 1974). Similarly, (Gupta and Govindarajan 2000) use "lateral" linkages in the same vein as horizontal relationships where they define lateral socialization mechanisms as those between "peer" nodes or units, such as horizontal personnel transfers inside the same organization. The closest attempt at discussing lateral linkages for integrating knowledge separately from horizontal relationships is found in (De Boer, Van Den Bosch et al. 1999), who define lateral knowledge integration as coordination and communication channels that cut across lines of authority, with mechanisms such as liaison devices between individuals or groups. By this definition, lateral integration channels can be considered as those crossing intra-firm boundaries to connect different entities at various levels of the hierarchy, with the purpose of bridging gaps in tacit knowledge and expertise. Building on these insights, I define lateral linkages as specifically those channels linking programs and functions, where the latter supply new knowledge to programs through mechanisms such as people rotation and liaison devices as suggested in the literature. In a product development context, functional personnel are domain specialists with deep expertise and up-to-date knowledge, which makes lateral linkages under this definition more likely to be used for integrating new tacit knowledge held by individual experts. Lateral linkages are therefore most useful in the development of complex and high technology products due to the increasing breadth and depth of disciplinary knowledge required to develop such products as well as the need for more up-to-date knowledge in these new technology environments (Allen 2000).

With a more complete picture of the vertical, horizontal and lateral dimensions for knowledge integration, we can conclude that knowledge integration in complex product development takes place *simultaneously* along all three types of channels, but with varying emphasis based on the level of knowledge dependence between the different tasks being performed, which is reliant on the level of dependence between the different parts of the product under development. Thus, in order to determine the paths and mechanisms for knowledge integration in a particular product development context, it is important to look at the characteristics of the product under development and those of the problems encountered during development as surrogate measures

[9] Merriam Webster defines "lateral" as *directed toward, or coming from the side*, while "horizontal" is defined as *directed toward individuals or entities of similar status on the same level.* It is common to see the two concepts used interchangeably in organization research – see for example (Tushman and Nadler, 1978) and (Hansen, 2002). I classify lateral linkages (such as the links between programs and functions) separately from horizontal links (as those between two programs or two teams in the same program) and independently of hierarchical (vertical) order.

of the knowledge dependence between different production stages and tasks. The relationship between the knowledge integration process and the characteristics of problems and products will be reviewed in detail under § 2.2 and § 2.3 respectively.

2.1.8 The Firm / Network Dimension

The knowledge-based view of the firm (Kogut and Zander 1992; Grant 1996a) argues that firms are more efficient than markets at integrating tacit and explicit knowledge due to their collective coordination and communication mechanisms (i.e. the infrastructure) and a unifying organizational culture (e.g. shared values, goals and vision) that fosters collaboration. It then naturally follows that a network of several firms with a shared purpose would have more knowledge and more integrating mechanisms at its disposal than a single firm and hence would be superior at integrating knowledge than a single firm. Such a network would be different from a collection of traditional buyer-supplier relationships which are typically vertical, one way and at arm's length. Instead, an inter-organizational network is a collection of "peers" where relationships are many-to-many instead of one-to-many, and where network flows can involve technology and know-how exchanges, joint product development activities, cooperative research, and collaborative marketing arrangements, among others (Grant and Baden-Fuller 1995).

However, there are many barriers that prevent networks from openly integrating their knowledge resources for the collective good of all members, most important of which are the proprietary barriers designed to protect each firm's knowledge from being imitated by outsiders. In a study of Toyota's high-performing network, (Dyer and Nobeoka 2000) identified three major dilemmas facing knowledge sharing in a network setting as: 1) proprietary barriers, 2) free-rider problems and 3) network infrastructure issues. The authors outlined four main strategies, practices and mechanisms that the Toyota Group uses to overcome these dilemmas, namely: 1) a network level association for sharing information through regular meetings, mutual training and socializing events; 2) a coordinating unit responsible for knowledge acquisition, storage and diffusion through free on-site assistance to network members, 3) sub-network level forums for specialized knowledge sharing in small groups of members, and 4) inter-firm employee transfers. The combined strategies and mechanisms establish a versatile infrastructure of multi-lateral

relationships between all members, providing each one with superior benefits from participating in the network, thus incentivizing the members to overcome their silo mentality maintained by each one's proprietary issues. These strategies and mechanisms also foster norms of reciprocity between members where being helped with a problem is contingent on one's commitment to helping others.

At first glance, one is tempted to believe that the Toyota model is based mostly on fostering "good intentions" between members, as perhaps best illustrated by the motto of the Four Musketeers "all for one and one for all". However, underlying this collaborative environment is a carrot-and-stick approach by Toyota where financial and other penalties are enforced against members who do not abide by the rules and norms of the network. This indicates that a strong shared identity and purpose among network members is not as easily implemented as in the single firm. As such, a network of organizations cannot be considered a true peer-to-peer arrangement like the open sharing "P2P" networks linking users of modern computers, rather there is always a need for leadership by a prime organization to mediate, facilitate and oversee the network to some degree, depending on internal and external factors such as network architecture and market forces for example (Gomes-Casseres 1994) (see § 2.2.4 in this thesis for a review of the characteristics of problem solving networks and § 2.3.5 for a review of the knowledge integration properties of different network structures).

Inter-organizational networks are increasingly common in product development due to the increasing demands for quality and capability in even the simplest of products, which translates to an increasing need for a wide variety of discipline-specific knowledge along with the need for deep specialization in multiple knowledge domains. As a result, organizations are less and less able to find all the knowledge they need within their own walls, and are faced with growing reliance on outside suppliers to provide the specialized knowledge required for the design and development of different parts of their product (Prencipe 2000). This is especially true in the development of complex systems where the required technical knowledge and expertise are more and more dispersed across large-scale networks of multi-tiered suppliers. This means that knowledge integration in complex product development is no longer confined to the walls of a single firm or to bilateral channels between prime and supplier, but can also encompass entire

networks composed of multiple organizations at different levels or tiers in the network. It then follows that the framing of knowledge integration in a complex product development context cannot be firm-centric only, and needs to account for the different strategies, practices and mechanisms that are useful in a network context.

Finally, it is important to note that while there are tangible benefits from network participation such as increasing performance in terms of increased quality, productivity and reduced inventory as has been shown in previous research (Dyer and Nobeoka 2000), there are nonetheless disadvantages inherent in network arrangements, such as knowledge dependence by smaller members on the lead or central firm in the network (Gomes-Casseres 1994), or rippling problems from one or more parts of the network which end up affecting other member firms directly or indirectly (e.g. if a new member upsets the balance of internal competition with an existing member, the benefits from network participation would decline not only for the two affected members but possibly for the entire network due to the decline in performance by two of its members).

The main insight from the firm / network distinction in the literature is that while the single firm is generally most efficient at integrating tacit and explicit knowledge within its walls, it is nonetheless not as effective at knowledge integration as a network of multiple organizations tied together through a strong shared identity, norms and rules.

2.1.9 The Direct / Indirect Dimension

(Hansen 2002) characterizes knowledge integration channels as either direct or indirect, where direct channels are those that provide immediate access to knowledge without going through intermediate connections, whereas indirect channels are those that go through intermediaries, such as boundary spanners or gatekeepers, in order to access knowledge from another source. The difference between direct and indirect channels is in the relative degree of their usefulness in the knowledge integration process, both in terms of efficiency (speed and cost of integration) and effectiveness (ease of absorption and relevance of the integrated knowledge to the problem).

Based on the concept of absorptive capacity advanced by (Cohen and Levinthal 1990), it is argued that direct channels between product development teams are most efficient for transferring tacit knowledge which is difficult to articulate and therefore more difficult to absorb than explicit knowledge. In such cases, effective mechanisms are individual face-to-face interactions or team meetings where knowledge about new technologies or product-specific technical know-how are quickly and more easily articulated and transferred from the source to the recipient. Indirect channels in such cases are only effective for identifying potential knowledge sources, but they are considered ineffective for integrating tacit knowledge due to the potential of distortion by intermediaries as they interpret the knowledge between source and recipient. (Hansen 2002) demonstrates that 1) the more direct channels a product development team has for integrating tacit knowledge, the more efficient they will be at accomplishing their task, and 2) the less intermediate connections an indirect channel has to go through, the more efficient the team will be in acquiring knowledge to accomplish their task. It is important to note here that the direct/indirect distinction pertains only to tacit knowledge integration, since codified knowledge can be readily integrated through conduits in the organization's infrastructure such as information systems and documents (as mentioned previously in § 2.1.2 and § 2.1.4).

Another perspective in the literature on the direct / indirect dimension is briefly discussed by (De Boer, Van Den Bosch et al. 1999) where knowledge integration is characterized as a process that can be directly or indirectly accomplished. Direct knowledge integration is defined as pre-designed integration where the expected outcome is pre-determined, such as integration using systems, manuals and policies specifically designed by the organization to accomplish a certain level of explicit knowledge integration. In that sense, direct is synonymous with *directed* integration. Indirect integration is defined as a guided (as opposed to directed) process where the outcome of integration is not pre-determined, and which involves autonomous agents, such as in the formal training and education of personnel or the establishment of formal liaison devices.

In summary, I conclude that tacit knowledge integration is most efficiently accomplished by establishing several direct channels between product developments teams, supplemented by a social network of indirect channels with short paths lengths (i.e. few intermediaries). Similarly,

explicit knowledge integration is best accomplished through directed channels and mechanisms pre-established by the organization, supplemented by formal training of personnel.

2.1.10 The Syntactic / Semantic / Pragmatic Dimension

Since knowledge used in production is information supplemented with context and experience (Nonaka and Takeuchi 1995; Grant 1996a), it is by definition embedded in its context, such as technical know-how embedded in a particular practice (as manifested in communities of practice (Brown and Duguid 1991)), or knowledge that is technology-specific or product-specific (Henderson and Clark 1990). This means there are contextual boundaries separating different knowledge domains, which adds a new dimension to the knowledge integration process in that it necessitates the use of special types of mechanisms to interpret and transform knowledge before it can be transferred across different knowledge boundaries. These mechanisms are known as "boundary objects" and "liaison devices" and serve to establish a shared context across boundaries (Star 1989; Carlile 2002; Carlile 2004).

There are three types of knowledge boundaries in product development, the syntactic (pertaining to differences in syntax or language), the semantic (relating to differences in interpretations) and the pragmatic (involving differences in functional specializations and interests) (Carlile 2002; Carlile 2004). There are four categories of boundary objects that map to the three different types of knowledge boundaries: 1) database repositories that provide a shared syntax for *transferring* knowledge across syntactic boundaries, 2) standardized forms and methods that provide a shared format for *translating* knowledge across semantic boundaries, 3) models (such as drawings, prototypes and computer simulations) for *negotiating and transforming* knowledge dependencies across pragmatic boundaries, and 4) maps (such as scheduling charts, process maps and workflow diagrams) for *representing and clarifying* knowledge dependencies across pragmatic boundaries (Star 1989; Carlile 2002). The third and fourth categories of boundary objects are considered to be of similar nature and purpose and are often combined together (Carlile 2002). In addition, while most boundary objects are distinct in terms of their nature and purpose, they are nonetheless complementary in terms of their usefulness for knowledge integration in that using one mechanism can serve to support the effectiveness of using another (e.g. using models

and maps can enhance the content of shared repositories, and vice-versa). Furthermore, each object can be useful across more than the one type of boundary it is mapped against. For example, all boundary objects are considered useful in mediating shared syntax, even if not as effectively as repositories are (Carlile 2002).

The main insights from this literature are that differences in knowledge contexts constitute different types of knowledge boundaries that require particular types of mechanisms for mediating knowledge across them. As such, it is important for organizations to have a portfolio of boundary objects at their disposal in order to efficiently and effectively integrate knowledge in different environments. This is particularly important in the large-scale development of complex systems where the knowledge that is embodied in these systems is increasingly diverse (from different disciplines) and specialized (from different practice domains), and where more organizations with different lingo, interpretations and interests are involved in the development process (Carlile and Rebentisch 2003).

2.1.11 The "Sticky" / "Leaky" Dimension

Knowledge in organizations is said to have a "sticky" characteristic (Von Hippel 1994; Szulanski 1996), in that it is difficult to integrate (costly to transfer, share and use) between source and recipient. This is due to the fact that knowledge most relevant for production is mostly tacit (as discussed in § 2.1.2), and that differences in syntax, interpretation or interests may pose a barrier to efficient and effective integration (Cohen and Levinthal 1990; Carlile 2002). But it is also argued that knowledge has a "leaky" or "mobile" characteristic (Hoopes and Postrel 1999), which is the opposite of "sticky" in that it's easy to lose proprietary knowledge across porous external boundaries with competitors. (Brown and Duguid 2001) explain this dichotomy by noting that stickiness is triggered by the internal division of labor inside large-scale organizations, where internal boundaries make it hard to transfer knowledge between different communities of practice (e.g. between engineering and manufacturing), whereas leakiness is triggered by the unifying effect of the external network that the organization is part of, since networks by definition unify different organizations with a core of common practices.

(Von Hippel 1994) outlines five different strategies for integrating sticky knowledge, as follows: 1) moving the required knowledge to where the problem or task is located – described by Von Hippel as "visiting the plant"; 2) moving the problem or task to where the required knowledge is located – described as "relocating the plant"; 3) iterating between multiple knowledge sites if the required knowledge is located at more than one site – described as "plant-to-lab and lab-to-plant trips"; 4) partitioning the problem or task into sub-problems or sub-tasks that each draw on only one locus of sticky knowledge – described as the "Firm X – Firm Y partition"; and, 5) reducing the stickiness of the required knowledge, described as "tacit-to-explicit knowledge conversion…using expert systems and…computer databases".

Expanding Von Hippel's illustration of the five knowledge integration strategies into actual mechanisms as per the main objective of this research, it can be inferred that the implementation of each of the above strategies in a complex product development context would require the following types of integration mechanisms, respectively: 1) site visits, co-location, liaison devices, boundary objects, taskforces, team meetings, people transfers and dispatching of subject matter experts to move knowledge to the problem locus; 2) co-location, prototypes and simulation (both of which can also be considered boundary objects) and off-site (laboratory) testing to move the problem to the knowledge site; 3) shared databases and integrated design tools to reduce the stickiness of knowledge; 4) same mechanisms as in option 2 to iterate between knowledge sites; and 5) same mechanisms as in option 1 to partition the problem, but used across intra- and inter-firm boundaries. In a large-scale complex problem solving context, the required knowledge is likely to be located at more than one site, therefore strategies 2) and 3) are less efficient and/or effective due to the cost and difficulty of moving the problem to multiple knowledge sites and iterating between them to solve the problem.

Other important mechanisms for overcoming knowledge stickiness are those that facilitate the mobility or leakiness of knowledge, meaning those mechanisms that are used to integrate knowledge across external boundaries (including prime-supplier, program-program or program-function boundaries), namely networks of practice (or communities of communities of practice) and the social networks of individuals, both of which serve to establish a common knowledge

base between people regardless of their location within the same or different organizations (Brown and Duguid 2001). These insights can be reframed as in Table 3 below:

Table 3: Knowledge Integration over Insulated and Porous Boundaries

Knowledge Characteristics	Boundary Characteristics	Knowledge Integration Characteristics
Leaky	External porous (e.g. program-function)	Communities of practice, networks of practice, social networking
Sticky	Internal insulated (e.g. program-program)	Job rotation, moving experts, shared or integrated systems, liaison devices, boundary objects, team meetings
Sticky	External insulated (e.g. prime-supplier)	Site visits, co-location, taskforces, shared systems

The usefulness of the sticky / leaky distinction in this context is in pointing to the counterintuitive role of organizational culture in segregating knowledge inside the one organization, while unifying it across different organizations. This is in contrast to the common wisdom which suggests that culture ties all the members of an organization together through a shared vision and beliefs regardless of their practice, whereas cultural differences across organizations separate even those individuals who share the same practice. However , from the perspective of integrating knowledge, when considering that a technician and a systems engineer in the same organization have little shared knowledge in common, whereas systems engineers in different organizations have a lot of knowledge in common, it becomes more apparent that organizations cannot be considered as single communities of practice tied by the organization's culture, they are in fact a collection of many and often distinct communities of practice with different knowledge contexts and different sub-cultures (i.e. the organization from this perspective is a community of communities (Brown and Duguid 1991)). In that sense, the knowledge integration process should not only be concerned with bridging differences related to the nature of knowledge itself (such as differences in syntax between differently specialized teams), but also with bridging the internal compartmentalization of practice inside the same organization, such as between different programs in a single firm, or between teams at different levels in the same program (e.g. a subsystem level team versus a system-level team).

The main conclusion from this literature for the purposes of this research is that knowledge can be sticky even within the smaller confines of a single program or firm; therefore it is important to recognize that knowledge should be integrated across both internal and external boundaries, and that external boundaries are porous both ways, such that a protectionist policy against leaking knowledge to the larger external network of practice is counterproductive as it will inhibit the reverse integration of knowledge from the network to the firm across those same boundaries. Instead, instituting a strong identity and shared purpose at the level of the network of practice that the organization is embedded in can leverage the leakiness of knowledge and enhance its integration.

2.1.12 Summary of Insights on Knowledge Integration

In this section I reviewed the main characteristics of knowledge in organizations and their influence on the knowledge integration process, specifically in a large-scale complex product development context. The previous sections highlighted the major insights and issues discussed in the knowledge integration literature and that will be accounted for in this research. Following is a summary of the main knowledge integration channels, mechanisms, strategies and practices that were outlined in the previous subsections, presented in Tables 4 and 5 below.

Table 4: Knowledge Integration by Knowledge and Organizational Characteristics

Knowledge Characteristics	Org. Boundary Characteristics	Knowledge Integration Characteristics* (* Only the primary channels and mechanisms are shown)
Tacit	--	Face-to-face communication, group interaction
Explicit		Documents, information systems
Component	Prime-supplier	Documents, information systems
Architectural	Intra-prime	People transfer (inter-project rotations)
Sticky	Internal, External	Site visits, co-location, liaison devices, boundary objects
Leaky	External	Networks of practice, individual social networks

Table 5: Knowledge Integration by Organizational Characteristics

Org. Boundary Characteristics	Knowledge Integration Characteristics* (* Only the primary channels and mechanisms are shown)
Syntactic	Database repositories
Semantic	Standardized forms
Pragmatic	Models/prototypes, drawings, simulations, maps
Vertical	Integration between subsystem-system teams, prime-supplier
Horizontal	Intra-program, program-program, peer-peer integration
Lateral	Program-function integration
Formal	Teams and taskforces, liaison devices, meetings, information systems, boundary objects, mediators
Informal	Personal networks, online communities of practice
Direct	Team or individual meetings face-to-face
Indirect	Social or organizational networks
Firm	Organizational culture, infrastructure
Network	Network identity, facilitator groups, rules

2.2 Insights from the Literature on Problem Solving in Complex Product Development

Knowledge is to problem solving as science is to engineering. In fact, the terms can be used interchangeably (and they often are). This is because the vast majority of productive human activities such as engineering and manufacturing can simply be described as problem solving, and all problem solving requires knowledge in the form of scientific and/or non-technical information, know-how and skills[10]. Knowledge is therefore the basis of problem solving, and its integration is what enables organizations to solve complex problems where multiple sources and types of knowledge are needed. This argument is supported in the knowledge-based view of the firm where the central proposition is that organizations exist because they are more efficient than markets at integrating specialized knowledge in order accomplish tasks and solve problems (Demsetz 1988; Grant 1996a; Nickerson and Zenger 2004). It naturally follows then that the two processes, knowledge integration and problem solving, are inextricably tied together such that the former enables the latter (i.e. solving problems is done by finding, acquiring and applying the necessary knowledge) and the latter is the reason and the vehicle for doing the former (meaning knowledge is integrated in order to solve problems and in the course of solving problems).

Therefore, I argue that problem solving is the thread through which the investigation of the knowledge integration process can best be accomplished. As a result, and while the process of organizational problem solving itself is not the main focus in this thesis, it is nonetheless important to understand the primary characteristics of problem solving in order to determine how they affect the choice of strategies, practices, channels and mechanisms for integrating knowledge in different problem situations. For example, several factors influence the level of difficulty of the problem solving task, including the difficulty of the problem itself, as well as the difficulty of coordination and communication with the parties involved in the problem solving process, among others. It is thus imperative to understand both the characteristics of common problems and the environment in which the problem solving process takes place. These will be explored in detail in the following sub-sections, starting with an overview of the related literature provided in Table 6 below.

[10] For example, one reason a problem is considered as such is because there is uncertainty regarding the action to take. Problem solving thus begins with a search for information that reduces uncertainty regarding what should be done and how, and ends with applying new information, know-how and skills in actually solving the problem.

Table 6: Overview of the Literature on Problem Solving in Complex Product Development

erence	Focus	Method	Relevant Conclusions/Results
& Bar- 004	Characteristics of complex problem-solving in large-scale engineering networks	Modeling	Problem solving networks in complex product development sparse (few connections) and highly clustered (short connect with more information flowing out of a node than into it
: o, 1991	Integrated (joint) problem solving as enabler of product and development performance	Conceptual & Empirical	Integration through face-to-face communication (cross-funct and cross-firm), mutual trust/commitment, shared responsibi
: Sproull,	Source-recipient relationships for transforming information into actionable problem solving knowledge	Hypothesis-test	Vertical channels for problem solutions (know-what, know-h referrals (other people, databases), validation, and legitimati Horizontal channels for problem reformulation
Chu,	The role of trust in building collaborative relationships and joint problem solving	Hypothesis-test	A lack of trust may cause suppliers to suppress design inforn useful for problem solving
Lengel,	Media richness for problem solving under uncertainty and ambiguity	Conceptual	High ambiguity problems require rich media e.g. group meet high uncertainty problems require IT coordination, data colle
rdt and 1995	The effect of technology maturity on product development efficiency and strategy	Hypothesis-test	An experiential, iterative problem-solving strategy is require new or rapidly changing technology environments, while a f structured approach is appropriate in mature environments
rd, 1998	The basic elements of the problem solving process for problems of high complexity	Conceptual	Three fundamental skills for complex problem solving: know (know-what), experience (know-how) and group communica
on & 2004	The relationship between governance choice and problem-solving approach/efficiency	Conceptual	Decomposable problems require low knowledge transfer and directional solution search, non-decomposable = high & heu
2002	The substitutability of specialist and trans-specialist knowledge	Conceptual & Modeling	An *efficient* problem solving team needs both design and integration knowledge, but only in a zero-sum amount
i, 2002	Knowledge partitioning and integration mechanisms across organizational boundaries	Conceptual & Empirical	Integrated problem solving and frequent communication with supplier improve design quality. For new designs, systems engineers need a high level of component-specific knowledg
ppel,	The locus of problem solving as a function of knowledge "stickiness"	Conceptual & Case-study	Knowledge used in problem solving is tacit and "sticky" (dif to transfer), as a result the problem solving process moves (o iteratively) to where the required knowledge is located

The preceding overview in Table 6 above provides an outline of the main themes and insights from the literature on problem solving as they relate to the process of integrating knowledge in complex product development. The main themes from this literature can be summarized as follows:

1) Different types of problems require different channels and mechanisms for integrating knowledge in order to solve the problem
2) Different problem solving contexts/environments require different channels and mechanisms for integrating knowledge
3) The knowledge integration process becomes more complex with increasing problem complexity
4) In complex product development, problem solving is most efficiently and effectively carried out by Integrated Product Teams (IPT's)
5) In large-scale organizational contexts, joint (or integrated) problem solving networks spanning team and firm boundaries increase task and product performance

The main themes outlined above will be elaborated on in the following subsections.

2.2.1 Knowledge Integration for Different Problem Types

Problems are generally viewed as challenges to be overcome or avoided, or perplexities to be figured out, or even difficulties to be endured. A problem is often described in terms of its most salient attributes such as how severe the problem is or how new it is. In common terms, problems are said to be hard or easy, big or small, new or old, interesting or trivial, clear or ambiguous, complicated or simple, among other general appellations. Problems are also often defined in terms of their causes (e.g. technical or social), their consequences (e.g. serious or minor) or even their locus (e.g. internal or external). In practice, a problem exists when there is a discrepancy or gap between required and actual results, along with uncertainty on how to close the gap (Nickols 1994). In the existing literature on problem solving, problems have been formally characterized as well-structured versus ill-structured (Simon 1973), well-defined versus ill-defined (Jonassen 2000), decomposable versus non-decomposable and nearly decomposable (Nickerson and Zenger 2004), routine versus non-routine (Mayer and Wittrock 1996), and

complex versus simple (Simon 1962). In the organizational literature, problems are also considered synonymous with tasks (Daft and Lengel 1986) to be accomplished under conditions of uncertainty (meaning the absence of knowledge and information) or equivocality/ambiguity (meaning the presence of conflicting knowledge and interpretation). It can be inferred that ill-structured problems are synonymous with uncertainty since more information is needed to arrive at a solution, whereas ill-defined problems are equivalent to ambiguousness due to the presence of conflicting information. A summary of the formal characterizations in the literature is presented in Table 7 below:

Table 7: Problem Characterization in the Literature on Problem-Solving

Problem Type	Problem Characteristics
a) Decomposable b) Nearly-decomposable c) Non-decomposable	a) Can be divided into independent sub-problems without tradeoffs b) Can be divided into inter-dependent sub-problems with tradeoffs c) Cannot be divided into independent sub-problems
d) Simple e) Complicated f) Complex	d) Unidimensional or consisting of a few independent dimensions/aspects e) Consisting of multiple dimensions/aspects that are not inter-related f) Consisting of multiple dimensions/aspects that are inter-related
g) Routine h) Non-routine	g) Can benefit from previous solutions h) Cannot benefit from previous solutions or requiring new combinations of previous solutions
i) Well-structured j) Ill-structured	i) Having specific initial conditions, goals and operators j) Having some unspecified aspects
k) Well-defined l) Ill-defined	k) Having a single, guaranteed solution l) Having multiple, non-guaranteed solutions

The usefulness of these formal characterizations is that they provide a starting point for describing a problem in generic and generalizable terms outside of specific contexts or disciplinary domains[11]. However, it is difficult to *fully* characterize a problem in terms of only one of its attributes. Thus, for example, a "well-defined" problem doesn't necessarily mean the

[11] Problems are typically characterized in specific and contextual terms that are not generalizable, for example health problems are typically defined according to the part of the human body that they affect (e.g. muscular problems, heart problems, chest pain, back pain...), often having to use specific terminology (i.e. medical jargon) to describe causes and symptoms of the problem. It is therefore difficult to make use of the medical terminology in order to characterize non-medical problems, such as legal type problems for example.

problem is simple or small; indeed it can be big or small, complicated or simple, old or new, and so on. This is why in order to completely characterize a problem it is necessary to address its multiple attributes including its scale, scope, novelty, structure, among others. It is also important to note that problem attributes are not all orthogonal to (or independent from) each other. For example, since problem complexity is defined in terms of the number of dimensions of the problem as well as their interrelatedness, it may be possible to measure problem complexity through surrogate attributes such as the scale and structure of the problem for instance. This strategy will be used to construct a measure of problem complexity for the purposes of this research, as will be outlined in Chapter 5 on "Methods for Data Collection".

But regardless of how a problem is characterized, what is most commonly associated with problems is action, meaning action in tackling the problem, and the associated knowledge required to act on the problem[12]. Problems can be fully described from the perspective of the knowledge required to tackle them, thus for example a problem described as "tough" in colloquial terms can be defined as requiring new knowledge or multiple sources of interdependent knowledge, and a problem described as "decomposable" in formal terms can be defined as requiring distinct and independent knowledge sets. Naturally, different types of problems require different types of knowledge to act on them, and when problems involve multiple interrelated issues (i.e. when the problem is complex), different *combinations* of knowledge may be required to tackle all aspects of the problem. Complex problems thus often require the combination or integration of multiple types of knowledge, which makes these problems the most common trigger of knowledge integration from multiple sources. This means that the investigation of the process of knowledge integration through the lens of problem solving necessarily involves a focus on complex problems.

In the product development context, complex problems are frequently encountered in the development of complex products due to the scale and interdependence of the various elements and component technologies that make up the product, thus these problems are by default the primary concern in this thesis since the focus in this work is on complex product development. Furthermore, this thesis is specifically concerned with development in a highly customized high-

[12] Merriam Webster defines "problem" as a question to be solved, and "question" as a test of knowledge.

technology environment where new unproven technologies (or new combinations of existing technologies) are often used for the first time, which gives rises to new problems unseen before. As such, non-routine problems are typical in this environment and will be at the center of the investigation in this work. Finally, since complex problems by definition involve multiple dimensions or variables that are "interrelated in non-trivial ways"[13], they are non-linear in nature such that they cannot be easily decomposed and tackled in a linear fashion; in other words addressing different aspects of the problem one at a time and independently of each other will not necessarily solve the entire problem due to the interdependence between them, and can give rise to unpredictable or undesirable results against single-point solutions. As such, non-decomposable problems make up the bulk of the problems being investigated in this thesis. Table 8 below presents a conceptualization of the different types of problems of concern in this thesis from the perspective of knowledge and knowledge integration:

Table 8: Knowledge Integration by Problem Type

Problem Type	No. of Knowledge Sources	Knowledge Interdependence	Primary Knowledge Integration Type
Decomposable	Low	Low	Explicit
Nearly-decomposable	Low	High	Tacit and Explicit
Non-decomposable	High	High	Tacit
Simple	Low	Low	--
Complicated	High	Low	Explicit
Complex	High	High	Tacit
Routine	Low	Low	Explicit
Non-routine	High	High	Tacit
Well-structured	Low	Low	Explicit
Ill-structured	High	High	Tacit
Well-defined	Low	Low	Explicit
Ill-defined	High	High	Tacit

[13] This definition builds on the principles of complexity theory in systems where a complex system is defined as one "made up of a large number of parts that interact in a non-simple way" (Simon 1962), as well as the principles of complexity theory in organizational design where large organizations are seen as complex due to the number and interdependence of tasks and stakeholders (Galbraith 1974).

Table 8 shows that problems where knowledge interdependence is high require mostly *tacit* knowledge integration whereas *explicit* knowledge integration is dominant for problems where knowledge interdependence is low[14]. This is based on insights from the existing literature (as summarized previously in Table 6) which state that decomposable problems require "low and directional" transfer of knowledge (Nickerson and Zenger 2004) as can be afforded efficiently in the exchange of explicit information between pre-determined parties. For example, in product development a purely decomposable problem maps to a perfectly modular product where the design problem for the entire product can be efficiently divided up into sub-problems (or design modules) and outsourced to suppliers, such that knowledge interactions become largely governed by bilateral exchanges of information between prime and supplier (thus the low number of contributing knowledge sources involved) in the form of requirements and specification information. This means that there is little need for the exchange of tacit knowledge to solve the complete problem, in fact the only knowledge that would need to be exchanged in this case is in the form of explicit information specifying the connection of each sub-problem to the whole (Baldwin and Clark 1997). It is important to note however that this is only true in the case of purely decomposable problems (as would be encountered in the development of a perfectly modular system), which is rarely the case in practice.

In reality, systems are not perfectly modular or integral but rather a hybrid combination of both strategies (Sanchez and Mahoney 1996). For example, while a desktop PC is typically a modular product in the sense that most components are standardized (such as the hard drive or the video card) and may be substituted without requiring a redesign of other components, it is nonetheless not a perfectly modular system since all components must function properly for the entire system to work. This means that there is some knowledge interdependence between different aspects of the design problem, however weak or simple they may be in this case, and so while the design of the desktop PC is a decomposable problem in theory, it is not a perfectly decomposable problem in practice. This is especially true and relevant in complex product design where modularity serves merely as a strategy to manage the complex knowledge interdependencies of the design without being able to completely eliminate these interdependencies.

[14] Note that this does not mean that only tacit knowledge is used in solving problems with high knowledge interdependence or that only explicit knowledge is used in solving problems with low knowledge interdependence, just that one type of knowledge is usually more dominant than the other.

In terms of knowledge integration, the development of a perfectly modular design can be mostly carried out using explicit knowledge interactions between the system integrator and the component developers in the form of design requirements, specifications and standards. However, in complex product development where modular architectures are not easily and completely decomposable, it is expected that knowledge integration cannot be restricted to explicit knowledge interactions only. In fact, since the modularization of complex product architectures must be done ex-ante without full knowledge or visibility into all aspects of the design, it is expected that extensive tacit knowledge interactions would be required to troubleshoot problems with incorrectly specified interfaces, requirements and standards.

In contrast, insights from previous research suggest that non-decomposable problems require "high and heuristic" transfer of knowledge (Nickerson and Zenger 2004) due to the interrelatedness of multiple aspects of such problems. In this case, numerous interdependent knowledge sets are needed in order to tackle the different aspects of the problem, and as such no single actor (or database) can possess all the knowledge required to solve the problem. In product development, a purely non-decomposable problem maps to a perfectly integral product where the design problem cannot be efficiently divided up into independent sub-problems. As such, there is no explicit information defining the relationships between different aspects of the problem as is possible with decomposable problems. This means that the knowledge required to tackle the entire problem is mostly tacit and distributed among multiple actors. The process of integrating knowledge in this case is highly iterative and often proceeds randomly until all the required knowledge is found, combined and applied to solve the problem. Knowledge integration in this case is more efficiently carried out internally to a single organization as it is more difficult to coordinate tacit knowledge across external boundaries, as already discussed in section § 2.1.7. However, the knowledge variety and diversity required to tackle non-decomposable problems typically forces the integration of knowledge from outside the boundaries of the one organization, making the knowledge integration process in the case of non-decomposable problems even more difficult and more involved. As noted above for decomposable problems, it is equally rare to encounter problems that are purely non-decomposable even in complex product development since products are rarely purely integral.

For example, while a notebook computer is considered an integral product in the sense that most components are fused together in order to increase performance (e.g. the video card is integrated with the motherboard to reduce weight, space, energy consumption...), and cannot be substituted without requiring a redesign of the entire system, it is still possible to design most of its components independently of each other (such as the display and the keyboard for example). This means that while the design of the notebook computer is a non-decomposable problem in theory, it is not a completely non-decomposable problem in practice.

In terms of knowledge integration, the development of a perfectly integral design requires tacit knowledge interactions to coordinate the interdependent design choices for different components since a change in one component design can affect other component designs in unpredictable ways. This phenomenon is known in complexity theory as emergent behavior and is characteristic of complex systems (Crawley, de Weck et al. 2004). Non-decomposable problems tend to be the most complex types of problems for the same reason that makes them non-decomposable (i.e. due to having multiple aspects that are interrelated in non-trivial ways, as explained previously in this section). This is why knowledge integration is identical for complex and non-decomposable problems in terms of the number and type of knowledge sources and iterations involved, as shown in Table 8.

In complex product development, most common are the nearly-decomposable problems which can be divided up into sub-problems that are not fully independent of each other, but where interdependence of knowledge sets across sub-problems is manageable with some tradeoffs. This is the case in hybrid modular-integral systems (i.e. systems that are not perfectly modular or perfectly integral) where design interdependence is managed by dividing up the system into modules which are developed independently by suppliers while still being tightly interdependent on each other's functionality, such that the final product assembled by the prime is a highly integrated design. In terms of knowledge integration, the development of hybrid modular-integral systems typically includes both the explicit knowledge interactions for dealing with decomposable problems and the tacit knowledge interactions for dealing with non-decomposable problems, to varying degrees. This is evident in the common division of labor between prime and supplier where module design is carried out independently by module suppliers and where

knowledge interactions tend to be restricted mostly to information coordination at "arm's length" (Clark and Fujimoto 1991), while final assembly is carried out by the system integrator and involves extensive tacit knowledge interactions among different teams responsible for the various modules making up the product.

And much like nearly-decomposable problems are somewhere between decomposable and non-decomposable, complicated problems are somewhere between simple and complex. However, complicated problems are not similar to nearly-decomposable problems in most cases. This distinction is illustrated in Table 8 by the fact that complicated problems involve multiple knowledge sources but weak knowledge interdependencies between different aspects of the problem, while the reverse is true for nearly-decomposable problems. Only in cases where interdependencies between different knowledge sets are trivial can a nearly-decomposable problem be considered as merely a complicated problem, as would be the case in the design of very loosely-coupled systems. In product development, the more loosely-coupled a system is, the more it is possible to develop its parts independently of each other and the more explicit the knowledge interactions can be during development (Sanchez and Mahoney 1996).

The knowledge integration process is also influenced by the structure of the problem, as shown in Table 8 for well-defined/well-structured versus ill-defined/ill-structured problems. The existing literature suggests that for problems that are ill-defined or ill-structured due to ambiguous knowledge about the problem (i.e. conflicting information and interpretations), a high level of tacit knowledge interactions are required (e.g. through group meetings) to resolve the ambiguity before tackling the problem. In cases of uncertainty (i.e. missing information), explicit information must also be collected and shared in order to tackle the problem (Daft and Lengel 1986). In contrast, well-defined and well-structured problems can be tackled with known procedures that are already captured and codified. These problems are most common in academic contexts such as the constrained problems that students encounter at the end of a book chapter and which require the application of explicit knowledge in the form of known concepts, rules and principles in order to solve the problem (Jonassen 2000). Simple problems are the most basic form of well-structured and well-defined problems and there is little need for integrating different types of knowledge whether tacit or explicit to tackle these types of

problems. This does not mean that simple problems are always easily solved, just that they don't require a combination of multiple sets of knowledge to be solved (Simon 1976). However, the particular knowledge or skill required to tackle a simple problem may not be resident in the organization, or these problems may be overlooked (e.g. given low priority) long enough to end up having big consequences. Such organizational factors that affect problem solving (regardless of the difficulty or structure of the problem) will be discussed in the following section § 2.2.2 on complex problem solving. Note that ill-defined and ill-structured problems are more common in practice than well-defined and well-structured problems which are more common in academic contexts (Jonassen 2000), much like complicated and complex problems are most frequently encountered in complex product development while simple problems are most frequently encountered in routinized development.

Finally, for routine problems, where previous solutions for similar forms of the problem already exist, there is little need for knowledge interactions beyond the access of explicit information from previous learning about the problem. Knowledge integration in this case consists mainly of aggregating explicit information for how to solve the problem and requires little tacit knowledge integration. Non-routine problems on the other hand are problems that have not been encountered previously. In complex product development, non-routine problems can be as common as routine problems as they typically result from emergent behavior that was unforeseen in the initial design, or as a result of implementing new unproven technologies or even new combinations of existing technologies in complex products. As a result, non-routine problems are typically complex problems that require the integration of tacit knowledge from multiple sources in order to deal with the interdependencies of the different aspects of the problem. Note that in multi-project organizations where there is poor resource planning in terms of allocating knowledge resources between projects, the frequency and complexity of non-routine problems increases dramatically and their occurrence can become a continuous self-reinforcing phenomenon known as "firefighting" (Repenning 2001).

In summary, this section provided an overview of the most generic and generalizable characterizations of problems as well as the relationship between different problem types and knowledge integration. The main insight from the related literature is that the more complex the

problem, the higher the number of contributing knowledge sources and the more tacit the knowledge that is involved in the integration process. In this work on complex product development, the focus is on non-routine complex problems ranging from the decomposable (most common in highly modular product development where the design problem can be decomposed into relatively independent sub-problems) to the non-decomposable (most common in the development of highly integral products where the design problem cannot be efficiently decomposed into independent sub-problems).

2.2.2 Knowledge Integration for Complex Problem Solving

Complex problem solving is most strongly associated with human thinking (Newell and Simon 1972), in fact the "complex" label in problem solving is commonly used in the literature on artificial intelligence (AI) to describe human reasoning and information processing, building on insights from such disciplines as cognitive science and human psychology with the purpose of endowing machines with capabilities similar to human intelligence. However, despite significant advances in computer science and robotics, automation of problem solving is still typically limited to simple and repeatable tasks since it is difficult to automate for complex tasks where unpredictable (or emergent) behavior is possible, as is typical of problems encountered in the design and development of complex products for example. The individual is therefore at the heart of complex problem solving, and it has been widely established that the more complex the problem the more the individual's experience becomes important in solving it (Jonassen 2000). As pointed to in the previous section § 2.2.1, problem complexity increases the scale of the problem solving process due to the high number of knowledge sources involved, as well as the difficulty of the search and reasoning tasks in a context of high knowledge interdependence. Therefore, complex problem solving typically involves more individuals with greater experience than routine problem solving. The most common example of complex problem solving in practice is in the design and development of complex products. What makes the problem solving process complex is the ambiguity of the problem statement (e.g. ambiguous customer requirements), the lack of a pre-determined solution path, the need to integrate multiple knowledge domains, limited or delayed feedback from the world (e.g. constrained prime-supplier relationships) and the "satisficing" nature of design solutions (i.e. answers that tend to be neither

right nor wrong, only better or worse, which makes it harder to reach conclusive solutions) (Simon 1973; Jonassen 2000).

In order to further frame the characteristics of complex problem solving as they relate to the process of integrating knowledge, it is important to first define the main stages of the problem solving process in an organizational context, namely to define how organizations go about solving complex problems in practice. While there is no formal definition in the literature for what constitutes organizational problem solving, it is broadly assumed that the process is synonymous with group problem solving and therefore involves typical tasks and activities that individuals carry out when tackling problems in a collective format. From the standpoint of human cognition, problem solving consists of search and reasoning to a) represent the problem and b) find a "satisficing" solution (Simon 1983). Expanding this definition and scaling it to the organizational context, I define organizational problem solving as a process of:

1) Identifying and defining the problem in terms of its consequences or possible causes
2) Diagnosing the root cause of the problem (or alternatively defining a desired solved state)
3) Searching for and formulating a solution (or alternatively a workaround from the current state to a pre-defined solved state)
4) Building organizational consensus around the solution or workaround (in terms of its feasibility, efficiency and effectiveness) and;
5) Applying and verifying the solution in practice

This definition is inclusive of the most typical stages of problem solving in an organizational context and is based on the most commonly performed tasks by individuals in the course of practice (Clark and Fujimoto 1991; Fujimoto 1999). I note here that in complex problem solving, the above steps are not followed sequentially or one at a time[15]. In fact, since problem solving is a cognitive process and since complex problem solving occurs in groups, it is more typical to see the steps of the process being carried out in random order and in parallel much like

[15] Complex problem solving tends to be an iterative process of trial and error due to the interdependencies between different aspects of a complex problem (Simon, 1962), and where solution development often goes through a refinement process until a better performing solution is reached.

collective brainstorming, often having to revisit the same step or having to work back to a previous step depending on information available about the problem as well as the knowledge available to tackle the problem. Therefore from the standpoint of knowledge integration, organizational problem solving by the above definition involves several knowledge processes which are carried out simultaneously (depending on knowledge availability) where individuals search for and access existing knowledge (both internally and from outside sources) about the problem and its possible solutions, brainstorm new knowledge if needed (e.g. to redefine the problem or to develop a new solution), and combine new and existing knowledge when required (e.g. for complex non-routine problems which cannot be solved entirely with existing knowledge). Other knowledge processes are employed when applying the solution and verifying it in practice such as testing and feedback for example. This implies that problem solving is enabled by the integration of knowledge about the problem and about the possible ways for solving the problem, as well as about the performance of the chosen solution. Thus, it can be argued that problem solving success is dependent on the efficiency and effectiveness of integrating knowledge at all stages of the problem solving process as defined above, and by the same token that the process of integrating knowledge is carried out differently in different problem solving contexts. For example, the current literature suggests that an experiential, iterative problem-solving strategy is required in new or rapidly changing technology environments, while a more formal structured approach is appropriate in mature environments (Eisenhardt and Tabrizi 1995). It has also been shown that problem-solving routines can get entrenched in an organization's culture and thus hinder its ability at creating and integrating new architectural knowledge (Henderson and Clark 1990).

As already mentioned earlier in this section, problem solving can be routine or non-routine, the latter being associated with complex problems and often referred to as "troubleshooting". Troubleshooting non-routine problems is an iterative process due to the interdependencies inherent in complex problems, while a structured problem solving approach is typical for routine problems since these have already been tackled or solved previously. It is often possible to troubleshoot a complex non-routine problem by reducing it to a routine problem that has been previously solved, thus allowing the use of similar steps from earlier solutions to solve the new problem (Simon 1962). However, in highly complex and large-scale development settings such

as high technology manufacturing firms, the frequency and scale of non-routine problems can turn the troubleshooting process into a continuous phenomenon synonymous with extinguishing recurring fires, which means non-routine problems are dominant in complex development settings. This continuous "firefighting" phenomenon is not only due to the high level of technical complexity of the problems encountered in this context, but also due to organizational complexities in terms of the number of stakeholders involved in problem solving and the interdependencies in stakeholder relationships (e.g. partner organizations on one development program who are competitors on another development program), all of which hinder the efficiency and effectiveness of knowledge integration and problem solving in this setting[16].

(Brown and Duguid 1991) argue that in the context of complex troubleshooting, knowledge does not come from what is taught in the classroom, but rather from informal story-swapping among technicians and users during practice. This means that knowledge most relevant for complex non-routine problem solving is the tacit knowledge and skills that practitioners develop with experience, whereas for routine problems the knowledge required for problem solving is more explicit since it has already been captured in previous solutions of similar problems. Furthermore, the current literature on knowledge integration suggests that individuals use different channels and mechanisms to integrate knowledge in routine versus non-routine problem solving situations as well as for different stages of problem solving (Daft and Lengel 1986; Cross and Sproull 2004). For routine problems, the required explicit knowledge in previous solutions or insights is most efficiently integrated using technologies such as expertise locators and databases that allow the systematic tracking down of previous knowledge or expertise directly relevant to the new problem; for complex non-routine problems where previous solutions are not directly applicable and where relevant knowledge is not known a-priori, the knowledge integration process involves a less orderly search for "help" from experienced individuals, which is best accomplished through the social network of the knowledge seeker. This network search enables the seeker to find and access a wide variety of sources for tacit knowledge outside their immediate circle and to actively engage these sources in the problem solving process. In these cases, qualitative and quantitative research have shown that individuals use horizontal channels

[16] In addition, the poor planning and allocation of knowledge resources adds fuel to the firefighting phenomenon especially in large-scale multi-program organizations where resource coordination involves tradeoffs between different programs (Repenning 2001).

during problem formulation and problem diagnosis since peers are more likely to become engaged in the early stages of problem solving than superiors higher up in the hierarchy, while vertical channels prove more useful for developing and validating solutions since senior experts and managers possess the deepest expertise and widest set of skills needed to provide actual solutions for complex problems (Cross and Sproull 2004).

In conclusion, this section has established that complex problem solving is an iterative process enabled by the integration of tacit knowledge through the social networks of individuals, with horizontal channels being most useful for problem diagnosis while vertical channels serve to develop and validate solutions.

2.2.3 Knowledge Integration in Problem-Solving Teams

According to (Mumford 1998), there are three fundamental skills for complex problem solving: knowledge (know-what), experience (know-how) and group communication. In this context, collaborative processes inside and across groups are considered key to problem solving because no one person embodies the breadth and depth of the knowledge necessary to comprehend complex problems, and because codified knowledge is seldom sufficient to deal with actual problems in practice. Complex problem solving is therefore typically carried out in a team environment as the most common vehicle for integrating the knowledge of multiple individuals around common tasks or problems. Teams are thus said to be the locus of complex problem solving in organizations[17], with heterogeneous teams being the most common form of group problem solving in this context due to the knowledge diversity they possess in terms of varied skills, experience and tacit knowledge (Walz, Elam et al. 1993).

Teams are typically formed around common tasks, such as subsystem design teams or system integration teams in product development, or around recurring types of problems such as quality teams in manufacturing. Even when teams are formed around products, projects, customers, geographic regions, functions or processes, the underlying basis for any team's formation is the

[17] (Nonaka 1994, p. 23) suggests that teams provide a "field in which individual perspectives are articulated, and conflicts are resolved", which can be interpreted to mean that teams are the locus of problem solving (i.e. conflict resolution) through the integration of diverse tacit knowledge (i.e. the articulation of individual perspectives)

ability to solve common problems in the most efficient and effective manner. This translates to the team having the ability to routinely integrate knowledge related to the most common problems it encounters, at the lowest cost in time and money. However, for complex problems it is typically necessary for teams to search for new knowledge sources in order to tackle the problem, which forces the team to go into a non-routine problem solving mode. In such cases, there is often a need to form special teams on a temporary or short-term basis to help or take the place of the regular team in order to solve a specific problem, such as the case of taskforces charged with troubleshooting non-routine and/or particularly difficult problems (Galbraith 1974).

In large-scale complex product development where complex problems are the norm rather than the exception, the complexity of the problem solving process is managed by forming teams at various levels of the organization (such as project or platform teams, product teams, component teams, etc...) to work on different parts or levels of the problem, thus creating a hierarchy of teams that mirrors the product architecture or the structure of tasks to be accomplished. The internal structuring of the teams themselves also affects their ability to search for and acquire new knowledge for problem solving, where team structuring involves decisions concerning leadership, membership, meeting frequency, and authority level in the organizational hierarchy (see § 2.3.6 for a detailed discussion of the effects of team structure on knowledge integration and problem solving). Problem solving teams typically meet daily or periodically to review and solve all problems within their mandate and capabilities, with the bigger problems typically referred upward in the hierarchy (i.e. through vertical channels). It is hypothesized that the greater the complexity of the problem or the uncertainty of the task that a team is tasked with, the greater will be the number of levels at which the team will operate, the more frequent will be its meetings and the greater its authority (Galbraith 1974).

More specifically and from the standpoint of knowledge integration, the channels and mechanisms utilized to integrate knowledge across team boundaries vary depending on the problem solving approach. In complex product development, team problem solving can be categorized along two main types of strategies: 1) Localized problem solving and 2) Joint (or integrated) problem solving. The former is internal to a program or organization and typically involves one or only a few teams, while the latter is across several teams (e.g. different problem

solving tasks are carried out concurrently by more than one team) and transcends organizational boundaries (Clark and Fujimoto 1991; Fujimoto 1999). The current literature on complex product development suggests that the most important enablers of integrated problem solving are face-to-face communication (both cross-functional and cross-firm), mutual trust, mutual commitment and shared responsibility (Clark and Fujimoto 1991). Hybrid organizational structures such as the matrix form represent a complete commitment to joint problem solving and shared responsibility[18] while the pure forms (such as the project or functional structures) lend themselves to more localized problem solving internally to the project or function respectively (Dosi, Hobday et al. 2000). However, since joint problem solving transcends organizational boundaries, the structure of the entire network is often a more important enabler of joint problem solving than the internal structure of the organization, as will be discussed in more detail in the following section § 2.2.4 on problem solving networks.

Joint problem solving is typically the more likely approach for dealing with complex problems since the number of knowledge sources involved increases with increasing problem complexity, thus the scale of the knowledge integration process expands beyond the localized setting. This also means that increasing product complexity necessitates more integrated problem solving since complex problems are more likely to occur in complex product development as outlined previously in § 2.2.1 on problem characteristics. However, the architecture of a complex product is another important driver of problem solving strategy and it is well established in the product development literature that complex problems encountered in the development of modular architectures are typically decomposable enough that they can be solved independently by different specialists under the supervision of a system architect as team leader (Ulrich 1995). Thus in a purely or highly modular case, problem solving would be localized and the integration of knowledge would be mostly internal to the team or along vertical channels between counterparts at different levels of organizational hierarchy (such as a subsystem and system team) or in different organizations (such as prime and supplier organizations). Conversely, the current literature suggests that integral architectures require joint problem solving by multi-disciplinary teams under the oversight of an experienced (or "heavyweight") system integrator

[18] Other hybrid organizational forms that emphasize lateral relations to varying degrees all utilize joint decision making and shared responsibility but not to the degree that a pure matrix organization does (Galbraith, 1974)

(Ulrich 1995), such that the integration of knowledge would require horizontal interactions between several teams within the same organization. These insights are summarized in Table 9 below:

Table 9: Problem Solving Teams in Complex Product Development

Problem and Product Characteristics	Problem Solving, Knowledge Integration and Team Characteristics
High complexity	Joint problem solving by large multi-disciplinary teams through direct communication, enabled by mutual trust and commitment
Modular system	Localized problem solving inside product team, between subsystem and system teams or between prime and supplier
Integral system	Joint problem solving between multi-disciplinary teams in the same organization

It is important to note here that this does not mean that modular product development cannot benefit from integrated problem solving, or that problem solving for integral products cannot be localized. In fact, research findings from the automotive industry show that for complex modular designs involving new technologies, integrated problem solving between the prime contractor's systems engineers (who are responsible for system integration) and supplier component engineers (who are responsible for component design) is required in order to solve unexplored engineering problems, and that frequent communication between prime and supplier improves design quality (Takeishi 2002). Similarly, it is suggested that problem solving in the development of purely or highly integral products is more efficiently conducted internally to the one organization since it is more difficult to coordinate tacit knowledge across external organizational boundaries (Christensen, Verlinden et al. 1999).

In reality, problem solving in complex product development requires a mixed approach between localized and joint problem solving since complex products are rarely purely modular or integral, and because a single firm or team is unlikely to have all the knowledge required to tackle highly complex problems (as discussed in more detail in § 2.2.1 and § 2.3.1). This means that the knowledge integration process in this context is likely to involve a combination of internal and external as well as horizontal and vertical channels but to varying degrees, depending on the

degree of complexity and modularization of the product under development. Similarly, the mechanisms of knowledge integration in this context are likely to be a mix of explicit and tacit mechanisms of varying types depending on factors related to the problem (e.g. technology novelty) and to the organizational environment (e.g. number of stakeholders), where the latter further dictates the structural characteristics of the problem solving team itself and its access to particular mechanisms for knowledge integration. As such, the literature suggests that an efficient problem solving team in a product development context needs both design and integration knowledge, though only in relative amounts as a system integration team only needs enough design knowledge to avoid architecting a system that is not producible, while a design team only needs enough system integration knowledge to avoid producing designs that present integration problems (Postrel 2002).

Finally, for both localized and joint problem solving strategies, it is increasingly common for third party experts to move from their normal workplace to the actual site of problem solving (such as joining a product team for example) in cases when problem complexity, ambiguity or uncertainty are high. This is because geographic proximity is a strong enabler of efficient and effective knowledge exchange as demonstrated by (Allen 1977), making "co-location" of project team members from different functions or organizations a frequent necessity. In complex problem solving where tacit knowledge integration between multiple locations is required, it is more efficient to partition the problem into sub-problems assigned to separate teams where each team has all the knowledge and expertise needed to solve their particular sub-problem, since it is typically inefficient for the problem solving team to iterate between multiple knowledge sites (Von Hippel 1994). This is most commonly observed in the development of complex products which are typically partitioned into subsystems or modules that are developed independently between prime and supplier organizations.

However, in cases of high complexity where final integration of the partitioned system is not trivial and typically gives rise to complex system integration problems, it is not uncommon to move knowledge from multiple locations to the locus of problem solving. For example, if a prime contractor faces system integration problems involving a subsystem designed by one of its suppliers, it is more efficient to co-locate a few supplier experts at the prime facility (i.e. with the

prime's system integration team) in order to integrate their design knowledge about their particular subsystem into the troubleshooting process. Conversely, if the problem is diagnosed to be rooted in the procured subsystem, it would become more efficient for the prime's product team to do "site-visits" to the supplier facility where all the design knowledge about the subsystem is located, since knowledge used in complex problem solving is mostly tacit and sticky (i.e. difficult to transfer independently of its holders), as already discussed in § 2.1.11. In reality, since complex troubleshooting is an experiential iterative process as explained above, individuals from both the prime and supplier organizations often have to shift repeatedly between different settings before they can reach an understanding of the underlying problem and develop possible solutions. In that sense knowledge is integrated by moving key actors from different teams across spatial barriers within and across organizations to diagnose various aspects of a problem and to develop creative solutions. In extreme cases, taskforces are used as a form of special teams grouping experts from multiple organizations and co-located nearest to the site of the problem.

To sum up, complex problem solving in organizations is most commonly performed in a team environment and jointly with other teams at various levels of the hierarchy as well as teams in other organizations. Depending on the characteristics of the problem and the location of knowledge resources, a team will integrate the knowledge it needs for problem solving by moving members from or to the team on a temporary or permanent basis.

2.2.4 Knowledge Integration in Problem-Solving Networks

As already discussed in the previous sections, organizational problem solving is a collaborative (team) process which requires access to multiple and diverse knowledge resources that are mostly experience-based (i.e. involving the tacit knowledge of individuals) (Almeida, Song et al. 2002). The number of individuals and teams involved in problem solving increases with increasing problem complexity, stretching from a small problem solving team of a few engineers from the same organization to a problem solving network (or networks) of thousands of technical and non-technical people from multiple organizations. In large-scale complex product development, the product is typically partitioned into smaller modules which are outsourced

externally in order to manage design complexity, thus creating a network of interconnected tasks that are handled by separate teams or individuals distributed across different partner and supplier organizations. This has the effect of decomposing the overall design problem into sub-problems and sub-tasks which require access to fewer knowledge sources, while at the same time increasing the number of knowledge resources for problem solving that each member firm can draw on from the larger network. However, previous research in this context has shown that the resulting problem solving networks are sparse (e.g. there are few connections between members of the network) and highly clustered (the connections are short), with more information flowing out of a node than into it (Braha and Bar-Yam 2007). The sparseness of the network can be attributed to many factors including the cost of establishing a connection with other nodes in the network or the limited capacity of each node for processing information due to bounded rationality (Simon 1973). Clustering means that problem solving in large-scale networks is distributed in nature mirroring the structure of the decomposed product, where each outsourced module leads to the formation of a small problem solving network near the supplier of that module. It is thus that joint or integrated problem solving between small clusters becomes increasingly necessary for coordinating the distributed problem solving activities of individuals and teams involved in the development of complex products (Clark and Fujimoto 1991). The channels for knowledge integration in this context are therefore not limited to inter-team connections only but also encompass links between clusters of teams or individuals.

Along similar lines, it is also the case that problem-solving complexity is not only driven by the difficulty of the problem itself but also by the environment in which the problem is tackled, such as the number of parties involved in the process and the relationships between them. For example, it maybe more difficult to solve a relatively simple problem if it involves stakeholders who are outside the boundaries of the team or the firm than to solve a more complicated or complex problem where all the knowledge required is resident on the problem solving team. For example, in a product development context, a major source of integration problems is due to module suppliers failing to communicate hidden design information to the integrator (Baldwin and Clark 1997), and it has been shown that it is in fact the lack of trust in the network that can cause suppliers to suppress design information useful for problem solving (Dyer and Chu 2003).

This situation often requires the use of mediators in order to integrate knowledge between network members with poor ties.

Furthermore, and as already explained in § 2.1.8 on knowledge networks, there are disadvantages inherent in organizational network arrangements such as knowledge dependence by smaller members on the lead or central firm in the network (Gomes-Casseres 1994). This is especially true in complex product development networks where the lead firm is the system integrator which has the most authority and deepest problem solving expertise (Brusoni and Prencipe 2001). This means that members of the network often have to rely on the lead firm for help in solving "tough" problems, which increases the complexity of the problem solving process due to the increase in the number of parties involved. Reliance on the lead firm also diminishes learning opportunities of the other members of the network. Along the same lines, member firms very frequently have to go through the lead firm in order to acquire knowledge from other members, which reduces the integrity of the problem solving process due to the distortion in knowledge integration across indirect channels.

Combining the existing insights above, the main conclusion for knowledge integration from this literature is that complex product development is typically carried out in large problem solving networks made up of smaller clusters, and which can be represented by numerous short connections inside each cluster with fewer arm's length connections between clusters. Furthermore, effective integration of knowledge along inter-cluster channels is contingent on relationships of trust which connect different clusters through direct connections without having to go through a "middle-man".

2.2.5 Summary of Insights on Problem Solving

In this section I reviewed the main characteristics of problems and complex problem solving in organizations and their influence on the knowledge integration process, specifically in a large-scale complex product development context. The previous subsections highlighted the major insights and issues discussed in the problem solving literature and that will be accounted for in

this research. Table 10 below presents a summary of the main characteristics of knowledge integration that were outlined in the previous subsections.

Table 10: Knowledge Integration by Problem and Problem Solving Characteristics

Problem Characteristics	Problem Solving Characteristics	Primary Knowledge Integration Characteristics		
		Where	What	How
Decomposable	Localized, Routine	Inter-organizational	Explicit	Boundary objects
Nearly-decomposable	Localized and Joint, Non-Routine	Intra- and Inter-organizational	Tacit and Explicit	Team meetings; Site visits; Liaison; Boundary objects
Non-decomposable	Joint, Non-Routine	Intra-organizational	Tacit	Team meetings; Liaison devices
Simple	Localized, Routine	N/A		
Complicated	Joint, Routine	Intra-team	Explicit	Documents, Info Systems
Complex	Joint, Non-Routine	Intra- and Inter-organizational clusters	Tacit	Team meetings, Taskforces, Liaison devices, Co-location
Routine	Localized, Routine	Intra-team	Explicit	Documents, Info systems
Non-routine	Joint, Non-Routine	Intra- and Inter-organizational	Tacit	Team meetings, Taskforces, Liaison devices, Co-location
Well-structured / Well-defined	Localized, Routine	Intra-team	Explicit	Documents, Info Systems
Ill-structured / Ill-defined	Joint, Non-Routine	Intra- and Inter-organizational	Tacit	Team meetings, Taskforces, Liaison devices, Co-location

2.3 Insights from the Literature on Organization Design and Complex PD Literature

In the previous two sections I reviewed the literature on knowledge integration and problem solving in order to determine the primary channels, mechanisms, strategies and practices by which knowledge is integrated in organizations, and how this process is influenced by differences in the types of knowledge being integrated and the types of problems being tackled. The literature reviewed above was mostly generic (i.e. not specific to any particular organizational context) and the intent was on providing insights related to knowledge integration that can be generalizable to most any setting, with some attempts at relating the investigation to the specific context of interest in this research, namely that of complex product development in large-scale organizations. In this section I explicitly review the complex product development literature from the perspective of knowledge integration in order to draw insights about how knowledge is integrated in this particular context, and how the product characteristics and the organizational environment affect the process of integrating knowledge.

Similar to all other organizational processes, knowledge integration is influenced by the characteristics of its organizational setting, such as the structure of the teams, organizations and even the organizational networks in which it is carried out. And since the focus of this research is on complex product development, I will also investigate how knowledge integration is influenced by the characteristics of the product, such as product complexity, architecture and the underlying technology imbedded in it. Therefore, the purpose of the following sub-sections is to highlight these main factors of influence on the mechanics of knowledge integration in order to account for them in the field investigation (e.g. by designing the research instruments to collect data about these factors). This will serve to frame the knowledge integration process under different problem, product and organizational characteristics.

I begin by reviewing some of the more seminal studies which have already explored the most relevant factors affecting knowledge integration in complex product development, as shown in Table 11 below.

Table 11: Overview of the Literature on Organization Design and Complex PD Literature

Reference	Focus	Method	Relevant Conclusions/Results
Argyres, 1999	Coordination through information systems as a means of reducing design complexity	Case-study	In large-scale complex systems development, unstructured tacit design knowledge is efficiently codified and integrated across boundaries through standardized/common design tools
Almeida et.al, 2002	Multi-national forms vs. alliances in cross-border knowledge building	Hypothesis-test	Successful design teams have the dual ability to make use of computerized design tools and libraries to integrate explicit info, and communication for the tacit know-how of seasoned engineers
Baldwin & Clark, 1997	Modularity as a design strategy for dealing with increasing system complexity	Conceptual & Case-study	A major source of integration problems is due to module suppliers failing to communicate hidden design information to the integrator
Browning, 1999	Integration of design teams (IPT's) in complex systems development	Conceptual	Mediation, information sharing through IT and meetings, co-location and interface documents as IPT integration mechanisms
Christensen et.al, 1999	System architecture as a driver of organizational and industry structure	Conceptual & Case-study	Modularity in products leads to modularity in organizations where prime-supplier relations are arm's length, teams more autonomous
Dosi et.al, 2000	Co-evolution of organizational forms and complex problem solving behaviors	Heuristics	Program-oriented organizational forms with strong external communication channels best enable complex problem solving
Eppinger et.al, 2004	Misalignment of organization and product architectures in product development	Conceptual & Empirical	Teams designing integral systems communicate more effectively across subsystem boundaries than modular system teams
Fine & Whitney, 1996	The influence of core competence and product architecture on outsourcing	Conceptual & Case-study	A key skill in complex systems engineering is the flow-down of clear, complete and stable requirements to competent suppliers
Henderson & Clark, 1990	The impact of architectural innovation on the firm's existing knowledge and problem solving capabilities	Conceptual	A change in system architecture requires new knowledge about system interactions and new problem solving strategies, information filters and communication channels
Sanchez & Mahoney, 1996	Modularity in product and organization as an enabler of knowledge management	Conceptual & Case-study	Modular systems require more external coordination of interface information with suppliers, integral systems require more internal cross-team coordination of both design and interface knowledge
Ulrich, 1995	The influence of system architecture on the management of the development process	Conceptual	Troubleshooting modular systems involves debugging module interfacing problems, troubleshooting integral systems involves tuning multiple parts of the system due to part interdependence

The preceding overview in Table 11 above provides an outline of the main themes and insights from the literature on complex product design and organization design as they relate to the knowledge integration process. The main themes will be elaborated on in the following subsections.

2.3.1 The Link of Knowledge Integration to Product Complexity

A complex system is defined as one made up of a large number of parts and/or technologies that interact in a non-simple way. In such systems, given the properties of the parts and the laws of their interaction, it is not a trivial matter to infer the properties of the whole (Simon 1962). This means that most problems encountered in the design and development of complex systems are themselves complex, and it is the central task of design engineers and systems engineers to figure out how to simplify and solve these problems. This typically involves the integration of multiple knowledge domains from several knowledge sources (Carlile and Rebentisch 2003). As already outlined in § 2.2.1, dealing with complexity requires extensive tacit knowledge interactions between experienced specialists, and in product development these interactions are centered on the coordination of interdependent design choices for different parts of the overall product design since a change in the design of one part can affect other parts in unpredictable ways.

Complexity in systems increases with the increasing number of parts (meaning subsystems, components and elementary parts), increasing part variety (i.e. systems with identical components are less complex than systems of comparable size whose components are all different) and increasing part interdependence (Simon 1976). In high technology settings, system complexity is also a factor of design and technological novelty, with new unproven components or technologies being more difficult to integrate into a product system than mature components or technologies that are well understood in terms of their functionality and their interactions with other parts of the system (Novak and Eppinger 2001). However, building on the same interpretations used to classify problems in § 2.2.1, in this thesis I consider a system to be complex only when there is a high degree of interdependence between its constituent parts regardless of any other attribute such as scale, variety or novelty (the impact of technology novelty on system complexity and knowledge integration will be discussed separately in § 2.3.7).

Otherwise the system is considered to be simple or complicated depending on the scale, variety and novelty of its constituent parts[19]. In other words, the presence of a high degree of interdependence is a prerequisite for complexity, which increases with the increasing magnitude of its other characteristics such as scale, novelty and variety. It has been argued in the literature that complex systems are more efficiently developed in-house since the costs of coordinating knowledge across external organizational boundaries are higher than internally within a single firm (Christensen, Verlinden et al. 1999; Novak and Eppinger 2001). However, this view implicitly assumes that all the knowledge required to develop a complex system is available in-house or that it can be developed/acquired for less than the cost of coordinating knowledge with an external source. In cases of high system complexity, this assumption becomes too simplified in that it does not take into account the largely experiential nature of the knowledge required to tackle complex problems (as already outlined in § 2.2.1 and § 2.2.2 on complex problem solving), which means that the depth of experience required in this case can only be developed over a long period of time, thus making it inefficient for a single firm to develop it all in-house. A similar argument can be made about the breadth of knowledge needed to develop complex systems in terms of the large number and wide variety of knowledge contributions from multiple specialty areas, making it unlikely for one firm to possess or acquire all the different types of knowledge required for developing the entire system (Carlile and Rebentisch 2003).

An alternative view in the literature argues that in highly complex product development settings, system complexity takes the form of hierarchy (Simon 1962) meaning a complex system is one that can be partitioned into interrelated subsystems which are themselves made up of major interacting components, and the most common strategy for managing complexity in this case is to outsource some or all of these major constituent elements to external partners or suppliers at multiple levels or tiers who are then responsible for the entire design and development process of their assigned subsystems or components (Baldwin and Clark 1997; Langlois 2002). This reduces the complexity of product design but drives an increase in organizational complexity (i.e. the number and variety of stakeholders involved in the development effort and the interrelatedness of stakeholder relationships), thus increasing the need for knowledge integration

[19] A system is complicated when the number, variety and/or novelty of its constituent parts and technologies are high, whereas a simple system is one where all three characteristics are low. The behavior of a complicated system is predictable (no interdependence) whereas a complex system can (and typically does) exhibit emergent behavior.

across organizational boundaries. From this perspective, increasing system complexity is positively correlated with outsourcing[20] (as opposed to the "in-house" view in the literature where increasing complexity is seen as negatively correlated with outsourcing). Therefore and by implication, an increase in system complexity also increases the scale and scope of the knowledge integration process to involve individuals and teams across organizational boundaries.

(Crawley, de Weck et al. 2004) provide an illustration of knowledge integration in the design of complex products made up of multiple parts which are interconnected at multiple levels. They argue that complexity makes the design problem harder to solve since it takes a long time for an engineer to learn all the visible and hidden interactions between the different parts of the system. They add that only the most senior engineers (which they estimate at about 10% of all technical employees in most organizations) are likely to have the knowledge required for the design of the overall system. It follows that tacit experiential knowledge is most relevant to the process of integrating knowledge in complex product development. However, since outsourcing major parts of a complex system inevitably involves the exchange of information about requirements and specifications, and since the latter are likely to change frequently during the development process due to the complexity of the design, it is typical for the knowledge integration process in this context to involve extensive interactions of both tacit and explicit knowledge within and between prime and supplier organizations.

(Brusoni, Prencipe et al. 2001) have further shown that product complexity increases the need for knowledge overlap between assemblers and suppliers at various stages of design and integration. This means that increasing system complexity typically requires an increase in the transfer of system knowledge from prime to supplier, as well as the transfer of design knowledge from suppliers to the prime in order to better specify subsystem interactions and ensure modules can be integrated successfully into the overall system.

[20] Note that the outsourcing in question here is different and separate from the common outsourcing of minor parts to lower tier suppliers which is typically decided based on efficiency considerations (i.e. the cost of make versus buy) rather than for complexity reasons, as the design of these smaller parts is typically well understood and standardized. This makes it possible to outsource their development while minimizing the likelihood of integration problems.

In summary, a product is said to be more or less complex based on the interdependence of its constituent parts and technologies along with their number, variety and novelty, with high product complexity driving an increase in the scale and scope of the knowledge integration process especially in terms of tacit knowledge interactions across organizational boundaries. Increasing complexity also has the effect of reducing the delineation between the competencies of the developing firms, thus driving the need for prime organizations to exchange more system knowledge with their suppliers and for suppliers to transfer more design knowledge with the system integrators.

2.3.2 The Link of Knowledge Integration to Product Architecture

A product or system architecture is the translation of the intended functionalities of the product into a physical layout (Ulrich 1995; Whitney 2004). The architecture of a system is defined early on in the design process by the system architect and his or her team through a highly dynamic and iterative string of interactions with: 1) the customer – in order to define the customer's needs and formulate the requirements that the product must fulfill, 2) the systems engineers – in order to translate the customer's requirements into functions that the product must accomplish, and 3) the suppliers – in order to allocate functions to physical components that will carry them out (Rechtin 1991; Maier 1996). The first step involves brainstorming meetings to translate, understand and formalize the customer's requirements, as well as negotiate compromises for feasibility when necessary. The output from this stage is successive drafts documenting customer needs and product requirements, until a final set of complete and consistent requirements is reached. The second step involves defining the different functions that the system needs to perform in order to meet the requirements, along with the possible technologies required to execute the design. The output from this stage is a series of conceptual designs in the form of drawings, models and/or feasibility reports displaying possible alternatives. The last step is a structuring of the system by mapping functions to physical components and assigning the design and development of each component to a supplier. The output from this stage is the system architecture charts documenting the layout of physical components and specifying the interfaces between them as well as performance specifications

and standards documents for the parts and for the whole[21]. The architecture and the interface specifications and standards are known as the "visible design rules" (Baldwin and Clark 1997).

The steps and outputs described above show that the process of architecting complex products is highly dynamic and iterative and involves a deluge of technical and non-technical knowledge interactions of a multi-disciplinary nature among several stakeholders (Eppinger 2002). It includes explicit knowledge integration through such mechanisms as documents and models as well as tacit knowledge integration through mediation and group meetings. But more importantly, the characteristics of the knowledge integration process have been shown in previous findings to be related to the type of architecture of the product being developed. At one extreme, when the product is partitioned into separate "chunks" or subsystem modules that can be designed independently by different suppliers, the architecture of the product is said to be "loosely coupled" and referred to as a "modular architecture", which requires extensive external knowledge interactions between prime and suppliers. At the same time, a modular architecture allows the system architect to "black box" the design and provide suppliers with only those design rules that are required to develop their particular subsystem, and to *hide information* from them about the rest of the system (Parnas 1972; Sanchez and Mahoney 1996; Baldwin and Clark 2000). This helps the prime organization (i.e. the system architect and integrator organization) to protect its architectural knowledge about the system from being diffused to suppliers and competitors (Baldwin and Clark 1997; Langlois 2002). By the same token, detailed design knowledge for each module developed by the module suppliers becomes hidden from the prime organization as suppliers seek to protect their design expertise and increase their competitive leverage in the market, therefore segregating design and integration knowledge across integrator-developer (i.e. prime-supplier) boundaries.

On the other hand and at the opposite end of the architecture spectrum, when the different parts of the product are closely interdependent and cannot be designed separately, the system architecture is said to be "tightly coupled" and referred to as an "integral architecture", which

[21] Distilled from "The Art and Science of Systems Architecting" (1998) – a course presentation by Dr. Mark Maier (The Aerospace Corporation). The tasks and outputs described here are only a high-level abstraction of the actual systems architecting process which involves more steps and produces more outputs at each step such as analyses of cost, risk, safety, and reliability.

requires close coordination between different specialists in order to design and develop the entire system as one integrated whole. Such close coordination is most effectively achieved through extensive knowledge interactions within and between multi-disciplinary teams and is therefore most efficiently accomplished "in-house" (i.e. within a single firm) (Ulrich 1995; Christensen, Verlinden et al. 1999) (see also § 2.1.6 and § 2.2.1). However, for products where the level of design complexity is high enough that no single firm has all the knowledge and physical resources needed to develop the entire system, the development cannot be entirely completed in-house (as already discussed in § 2.3.1 above) and the product is typically partitioned into "large chunks" or subsystem modules which are outsourced to suppliers and subsequently integrated at the end by the prime. The benefit of this strategy is that it allows the use of modularization to reduce and manage design complexity but without compromising heavily on the performance and customization achieved in a highly integrated system. This is because integration is achieved inside each subsystem module, whereas overall system interdependencies are reduced through modularization at the level of subsystem-to-subsystem interactions. This architecture is thus closer to a "hybrid" modular-integral form and therefore requires a combination of external knowledge interactions with suppliers during the design phase and internal knowledge interactions between subsystem teams during the system integration phase.

The characteristics of the knowledge integration process are thus different for different product architectures, such that modular system development typically requires more external interaction of explicit information with suppliers about design requirements, interface specifications and standards at the subsystem level, while integral systems require extensive interactions internally across teams for both design and integration knowledge at the system level (Sanchez and Mahoney 1996). Furthermore, the mechanisms for integrating knowledge are different in different architecture regimes. In the integral case, the team leader is typically an experienced system integrator with a high level of authority for coordinating the numerous knowledge interactions between teams through such collaborative mechanisms as team meetings and common design tools for the entire duration of the design process, whereas in the modular case the team leader is usually an experienced system architect with the responsibility of partitioning the system and ensuring early and full specification of the visible design rules between different subsystems in advance of outsourcing them to suppliers (Ulrich 1995; Baldwin and Clark 1997),

with subsequent knowledge interactions being centered around coordinating these design specifications through documents and information systems. These insights are summarized in Table 12 below:

Table 12: Product Architectures in Complex Product Development

Architecture Type	Knowledge Integration Characteristics
Modular	External between prime and suppliers
Integral	Intra-firm (intra- and inter-IPT)
Hybrid	Intra-firm during system integration External during design phase

It has also been shown that changes in the architecture of a system require the development of new knowledge (referred to as architectural knowledge about system interactions – see § 2.1.6) as well as new problem solving strategies, information filters and communication channels (Henderson and Clark 1990). In the same vein, (Bozdogan, Deyst et al. 1998) have investigated architectural innovation in new product development through early supplier integration, showing empirically how tapping tacit supplier-based knowledge can yield significant benefits in terms of cost, schedule and quality performance in developing new products. Additionally, several empirical studies have shown that teams working on interrelated subsystems (i.e. within an overall system architecture that is integral) communicate more effectively with each other than teams working on loosely coupled subsystems (i.e. within an overall system architecture that is modular) (Sosa, Eppinger et al. 2000a; Sosa, Eppinger et al. 2000b; Sosa, Eppinger et al. 2004). The main implication of these insights for the purposes of this research is that problem solving is more localized and knowledge integration is more difficult in modular system development (namely due to information hiding and lack of communication) than it is for integral systems, which reinforces earlier insights about the need for integrated problem solving and frequent communication in order to improve design quality in complex modular products (Takeishi 2002) as outlined previously in § 2.2.3.

To summarize, the main insights for knowledge integration from the literature on system architecting and architectures are two-fold: first, the channels and mechanisms utilized to integrate design and system knowledge vary in type and frequency of use depending on the architecture of the product. Second, and due to the iterative nature of the system architecting process regardless of product architecture, the exchange of information about requirements, specifications and standards makes up the bulk of tacit and explicit knowledge interactions at multiple levels in order to formalize customer needs and flow down requirements, either through documents, information systems and prototypes or through group meetings and mediation.

2.3.3 The Link of Knowledge Integration to Product Platforms

A product platform (also known as a family of products) is a collection of physical and knowledge assets that are shared by a set of products, where physical assets range from product parts to production lines, and where knowledge assets include people and processes (Robertson and Ulrich 1998). Commonality within a platform is therefore a deliberate strategy whereby an initial product is "spun off" into other similar or more differentiated products by modifying or customizing an initial product design, and by using a common set of physical and knowledge resources in the development process. Knowledge integration within a product platform is thus designed into the development process and is typically accomplished through a "platform team", which is a multi-disciplinary group of engineers and functional specialists in charge of design and development for the entire platform of products, supported by top management in the platform planning stages (Sanderson and Uzumeri 1995).

In an extension of the single platform concept, one of the foremost principles of Lean product development is to consider *all* of the firm's products as part of a portfolio of platforms where knowledge, technologies and other resources must be shared to varying degrees in order to maximize the return on any initial investment (Cusumano and Nobeoka 1998). This means that knowledge integration is not limited internally to the single platform but must be extended across other platforms as much and as frequently as possible, depending on the level of design commonality across these platforms. Knowledge integration across platforms of products is especially relevant in complex product development where some degree of commonality

between different designs, or at least between parts of the designs of different products, is almost inevitable due to the numerous components and technologies involved. This makes knowledge sharing across programs a key enabler of an efficient and effective development process.

In a large-scale development context, finding the required knowledge and transferring and sharing it is not trivial or straightforward, which makes the process of integrating knowledge a challenging but essential ingredient in multi-project organizations such as manufacturing firms in the automotive and aerospace industries. In a thesis on multi-project management in the automotive industry, (Nobeoka 1993) showed that when different projects share a technology or design between them, there is a need for inter-project coordination between project managers as well as inter-project cooperation between design engineers, both directly (engineer-to-engineer) and indirectly (mediated by the functional manager overseeing both products). This means that when different platforms share a high degree of design or technology commonality, it is important to integrate knowledge between them at different levels and both directly (platform-to-platform) and indirectly (through the organization's functional department). In the same study however, it was shown that increasing product complexity makes it very difficult to coordinate and cooperate across projects due to the difficulty in managing design interdependencies in these cases. Nobeoka cited people transfers, joint activities, integrators (or boundary spanners) and a systematization of the informal networks of engineers as the most effective strategies and mechanisms for inter-project coordination and communication. These insights are summarized in Table 13 below:

Table 13: Platform Dependencies in Complex Product Development

Dependency Type	Inter-Project Knowledge Integration Characteristics
High design commonality, low complexity	Direct (between project managers, between design engineers) Indirect (mediated by functional managers)
High design commonality, high complexity	Direct and indirect plus liaison devices (integrators, personnel transfer, joint activities, formal networking)

To sum up, the most important insight from this literature is that in the context of multi-program organizations, intra-organizational knowledge integration should account for channels across

programs/projects especially in cases of component commonality or design and technology interdependence between programs. In such cases, both direct program-to-program channels and indirect channels (going through the functional organization) should be accounted for, and should be supplemented with additional liaison devices depending on design complexity.

2.3.4 The Link of Knowledge Integration to Organizational Structure

The relationship between the structure of the organization and the structure of its knowledge integration process has not been explicitly addressed in the literature. However much of the link between these two concepts has been covered in the insights about organizational forms and their influence on communication (Hedlund 1994; Grant 1996b), coordination (Sanchez and Mahoney 1996; Grant 1996a) and knowledge management (De Boer, Van Den Bosch et al. 1999) inside the single firm. It is commonly argued in the literature on organization design that different organizational forms (e.g. matrix, functional, project and hybrid combinations) enable information communication, task coordination as well as the transfer and sharing of knowledge to different levels of efficiency and effectiveness (Galbraith 1974; Allen 1977; Grant 1996b; De Boer, Van Den Bosch et al. 1999). This is because different forms provide different types of infrastructure by which knowledge can be integrated, both in terms of information systems and facilities and hierarchical arrangements. In the emerging knowledge-based view of the firm, the organizational form is seen as a vehicle for providing a set of "higher level organizing principles" and a rich social context to support the integration of knowledge (Kogut and Zander 1992; Almeida, Song et al. 2002). Each form presents advantages and disadvantages for knowledge integration, for example the project-based form brings all the functional disciplines on board the project (e.g. in cross-functional teams) thus facilitating knowledge integration across different specialties; however it disconnects each deployed specialist from his or her knowledge base outside the project. In contrast, the functional form keeps specialists connected with their knowledge base, but at the cost of weak knowledge integration across different knowledge domains and with the project teams (Allen 1997).

Hybrid organizational forms and what are known as innovative forms (such as the N-form (Hedlund 1994) and the Hypertext form (Nonaka 1994)) are designed to balance the advantages

and disadvantages of the pure structures, but they also present difficulties of their own. This is why knowledge integration processes are often tailored to overcome the disadvantages of any particular organizational structure through additional capabilities such as shared information repositories, multi-project meetings, and liaison devices, to name a few. These insights are summarized in Table 14 below:

Table 14: Organizational Forms for Complex Product Development

Organizational Form	Knowledge Integration and Problem Solving Characteristics
Functional form (dispersed knowledge structure)	Strong cross-program but poor cross-functional knowledge flow Integrated problem solving across programs
Project form (silo'ed knowledge structure)	Strong intra-program but poor cross-program knowledge flow Integrated problem solving across functional disciplines
Hybrid and Matrix forms	Use of boundary objects to overcome weak links (liaison devices, shared information repositories, cross-boundary communication)

In addition, and building on the discussion in § 2.3.2 about the influence of product architecture on the knowledge integration process, it has been well established in recent perspectives on organization design that there is a positive correlation between product architecture and the structuring of organizations (as well as entire industry structures) in what has come to be known as the co-evolutionary view of product and organizational architectures, and which has attracted much attention recently both in the literature and in practice (Christensen, Verlinden et al. 1999; Brusoni and Prencipe 2001; Galvin and Morkel 2001; Sako 2003; Sosa, Eppinger et al. 2004). The central argument in this body of work is that the architecture of the product influences the structure of the organization developing it, or as it is more commonly stated that "organizations design products and products design organizations" (Sanchez and Mahoney 1996), such that modularization in the product architecture leads to de-coupling of tasks and relationships in the hierarchy of the developing organizations. This is because modularization of an artifact is equivalent to the decentralization of the development process whereby problem solving is distributed to more autonomous teams in different organizations (Christensen, Verlinden et al. 1999) (leading to more localized problem solving as discussed in sections § 2.2.2 and § 2.3.2), and as a result the knowledge required to produce different modules is dispersed across team and organizational boundaries. Therefore, the organizational structure governing the relationship

between developing teams and organizations becomes loosely coupled, thus mirroring the modularity of the product architecture. This raises the need for more intensive knowledge integration across organizational boundaries (i.e. between the prime assembler and module suppliers).

Conversely, the development of integral architectures is more efficiently accomplished within a single firm since it is easier and less costly to coordinate task interdependencies internally rather than externally across the boundaries separating different organizations (this again assumes that the product complexity is low enough that all the skills required for the complete development effort are resident in-house as already discussed in § 2.3.1 above), and as a result integral product architectures create an inward pull on the vertical channels in the organization's structure through extensive reliance on internal horizontal knowledge interactions (Christensen, Verlinden et al. 1999; Wissmann and Yassine 2005). This increases the intensity of knowledge integration across internal boundaries inside the single firm at the expense of external knowledge integration, especially for tacit knowledge interactions between specialists across functional disciplines. Horizontal knowledge integration in this case is most effectively accomplished through integrated multi-disciplinary teams having the broad skills necessary to tackle integral designs (Ulrich 1995). Furthermore, explicit knowledge integration is also made more efficient in this case since tightened hierarchies (i.e. organizational structures which are tightly integrated along vertical dimensions) are designed to maximize task coordination and minimize the need for extensive information communication (Sanchez and Mahoney 1996; Grant 1996b). The relationship between product architecture, organizational structure and knowledge integration is illustrated in Figure 6.

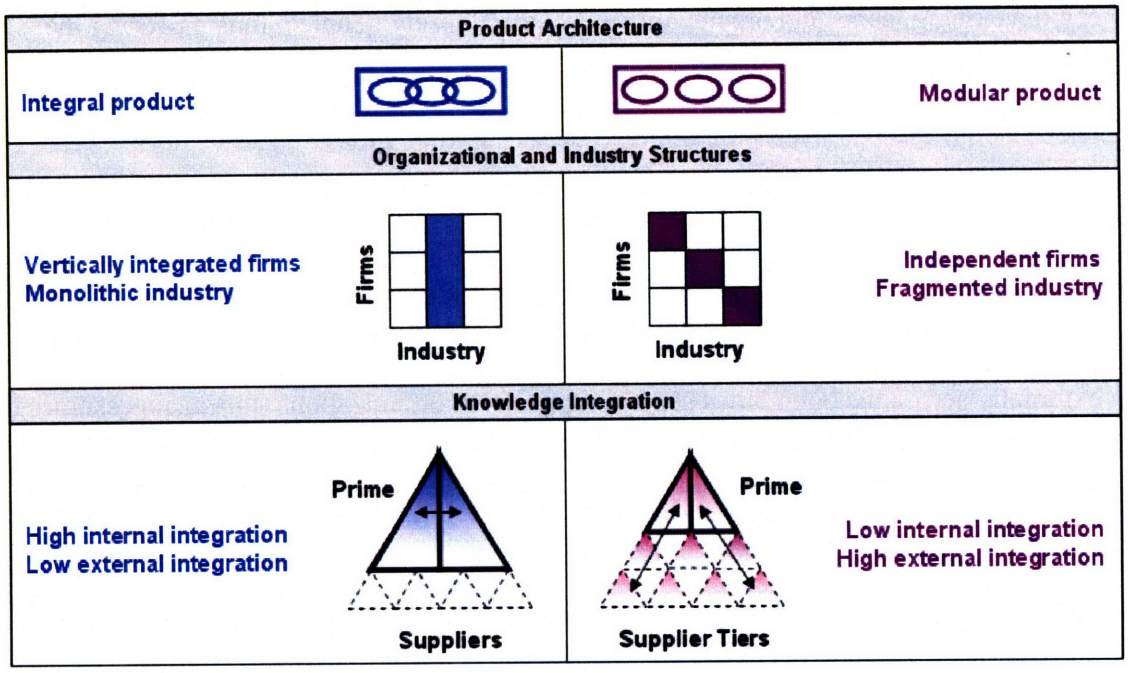


Figure 6: Product Architecture, Organizational Structure and Knowledge Integration

Comparing the two extreme cases above in the context of product development, it can be inferred that the main advantage of modular organizational structures where relationships are loosely coupled is in the well established (or formalized) and direct external channels with suppliers for the integration of explicit design knowledge (e.g. subsystem requirements and specifications information) since explicit knowledge integration does not require face-to-face interactions and can be accomplished efficiently through systems capabilities such as electronic systems and documents exchange (De Boer, Van Den Bosch et al. 1999). On the other hand, the main advantage of integral structures is in enabling close coordination of tacit knowledge about the design and architecture of the product such as individual know-how and experience in systems design, architecting and integration. This is done through well-established trust and commitment internally to the one organization and by informal channels and social networks that are not impeded by competitive barriers as is typically the case between different organizations.

It should be noted however that in reality organizations rarely adopt pure forms at one extreme or the other, just like actual products are rarely purely modular or integral (see § 2.2.1 and § 2.3.2).

In fact, it has been shown that the structure of organizations (and of entire industries even) oscillate periodically between the two extreme forms due to different technology and market forces (Fine 2000), and thus there is no single structure that dominates or endures indefinitely. Thus it is more common for organizations to adopt structures that are somewhere in between the two extremes and which are balanced in such a way as to overcome the disadvantages of any particular form, such as project-based organizations supplemented with close connections between projects and functions, as is most common in organizations developing complex systems (Dosi, Hobday et al. 2000).

Therefore, the main conclusion from this literature is that there is no single or universal type of organizational structure that most enables knowledge integration with maximum efficiency or effectiveness. Rather, for any given organizational structure there are fundamental channels and mechanisms that should be established to facilitate the efficient and effective integration of different types of knowledge across internal and external boundaries. The frequency of use of particular channels and mechanisms is then dictated by the product architecture at hand, among other factors such as the characteristics of the problem and the organizational environment.

2.3.5 The Link of Knowledge Integration to Network Structure

An extension of the relationship between the organizational form of the single firm and the knowledge integration process as discussed in the previous section is the connection with the structure of the larger organizational network (Grant and Baden-Fuller 1995; Sanchez and Mahoney 1996). Since this thesis is concerned with product development in a large-scale context, the influence of network characteristics on knowledge integration must also be considered. The knowledge integration and problem solving characteristics of complex product development networks have already been discussed in § 2.1.8 and § 2.2.4 respectively, with the main insights related to the structure of such networks being that they are "loosely coupled", such that clusters of suppliers are organized around a focal firm (the prime organization) and connected by channels that are both direct (peer-to-peer) and indirect (mediated by the focal firm). The main advantage of this structure is that it allows the focal firm to manage the design complexity of its products by outsourcing the design and production of some parts or

components across the network, while at the same time "hiding information" by keeping members in the network at enough distance to protect its knowledge of the overall system which is where its core competence lies, thus protecting its competitive advantage (Sanchez and Mahoney 1996; Baldwin and Clark 1997). However, this comes at the price of impeding knowledge integration across organizational boundaries due to the sparseness of connections between clusters. The reason for the emergence of this particular structure (i.e. loosely-coupled network) in complex product development is due to the strategy most widely adopted for dealing with complexity (namely the dominant modularization strategy for designing and developing complex products), which makes it more efficient for the prime organization to outsource production of different components and subsystems to the rest of the network (Prencipe 1997).

However, this picture is not static and recent literature has not provided any evidence of the emergence of a singularly unique or dominant network structure, particularly in rapidly changing technology environments, where recent research has revealed that changes in the product's underlying technology can lead to the emergence of different organizational arrangements as the focal firm tries to adapt by adjusting the structure of the overall network. More specifically, (Brusoni, Prencipe et al. 2001) identified three types of network structures based on the rate of technological change, namely: 1) tightly coupled, 2) loosely coupled, and 3) decoupled. Tightly coupled networks are needed to deal with fast changing technologies in highly integrated product architecture regimes, as they afford the focal firm the ability to quickly integrate changing technological knowledge with suppliers and to provide robust "total" solutions under conditions of high product complexity and high technological uncertainty. At the opposite end of the spectrum, decoupled networks are needed in stable technological regimes where the product architecture is highly modular as they allow the lead firm to efficiently integrate explicit knowledge about well defined standards and specifications. However, one of the drawbacks of decoupled networks is in the increased potential for "arm's length" relationships between prime and supplier organizations in the network as knowledge interactions become limited to flowing down design rules, thus making the vertical integration of tacit knowledge (when required) more difficult and prone to disconnects. In between these two extremes are the loosely coupled structures which afford the focal firm the ability to adapt to the transitional types of technology-architecture regimes (specifically, the stable-integral and unstable-modular regimes), allowing

the integration of tacit and explicit knowledge efficiently and effectively while protecting its own core competence (see § 2.3.7 for further discussion of the influence of technology maturity on knowledge integration). These insights are summarized in Table 15 below.

Table 15: Network Types for Complex Product Development

Network Type	Knowledge Integration and Problem Solving Characteristics
Tightly coupled	Prime can quickly/easily integrate new tacit and explicit knowledge from suppliers Integrated problem solving with suppliers (total solutions)
Loosely coupled	Prime can quickly/easily integrate old tacit and explicit knowledge from suppliers Cooperative problem solving with suppliers (protective of core competency)
Decoupled	Prime can quickly/easily integrate old explicit knowledge only Arm's length problem solving with suppliers (compartmentalized solutions)

To further characterize the types of channels and mechanisms by which knowledge is integrated under these different network arrangements, I draw on the structural attributes of each one as described in the literature. In the case of tightly coupled structures, (Brusoni and Prencipe 2001) describe tight coupling in terms of close, long-term relationships with suppliers for exchanging specialized knowledge, and (Sanchez and Mahoney 1996) classify those interactions as "intensive" and "hierarchical", which means that knowledge integration in this arrangement is mostly tacit (involving the specialized knowledge of individuals) and through vertical channels. Conversely, in the case of decoupled structures, coordination is said to be achieved through market mechanisms for exchanging knowledge about well defined standards and specifications. This means that explicit knowledge dominates the interactions in decoupled networks with the most efficient mechanisms in this context being information systems and technologies such as standardized/common design tools and databases (Argyres 1999). In loosely coupled structures, (Brusoni, Prencipe et al. 2001) propose that coordination is achieved through hierarchy and close cooperation in problem solving, while (Sanchez and Mahoney 1996) argue that coordinating the loosely coupled activities of component developers can be simply achieved through "information structures". Of course, since loosely coupled structures are appropriate in two diametrically opposed technology-architecture regimes as discussed above (i.e. the stable-integral and unstable-modular regimes), the choice of mechanisms for knowledge integration will depend on

which context the network is operating in. The previous insights are reformulated in Table 16 below.

Table 16: Network Types for Complex Product Development

Network Type	Knowledge Integration and Organizational Structure Characteristics
Tightly coupled	Tacit knowledge integration through vertical channels is dominant Tight long-term relationships with suppliers
Loosely coupled	Tacit knowledge integration through vertical channels is useful Explicit knowledge integration through information systems is dominant Flexible (close-to-loose) relationships with suppliers
Decoupled	Explicit knowledge integration through information systems is dominant "Arm's length" relationships with suppliers

In conclusion, there are three types of network structures in complex product development with the most common being the "loosely coupled" structure, where both tacit and explicit knowledge integration channels and mechanisms are necessary to varying degrees, depending on the rate of technological change and the architecture of the product under development. These networks require flexibility in the relationship between network members and the focal firm.

2.3.6 The Link of Knowledge Integration to Team Structure

A fundamental activity of teams is the integration of individual knowledge into collective knowledge in order to accomplish a task (Okhuysen and Eisenhardt 2002). Teams like organizations can have different structures (e.g. project teams, functional teams, matrix teams and hybrid teams), with each team structure having advantages and disadvantages for solving problems and accomplishing tasks, as well as varying levels of performance in different contexts. This is because the team's structure is a key determinant of its ability to integrate knowledge internally and externally with the environment. Some of the key determinants of team structure are team composition, size, leadership and level in the hierarchy (i.e. the level of authority in the organizational hierarchy at which the team operates) (Galbraith 1974; Brown and Eisenhardt 1995).

Team composition is an important aspect of team structure since it affects the team's ability at integrating knowledge both internally and externally. For example, it has been shown in previous research that successful design teams are composed of individuals who have the dual ability of using computerized design tools and libraries to integrate explicit information efficiently, as well as the right communication skills and social networks for effectively integrating the tacit know-how of seasoned engineers from outside of the team's boundaries (Almeida, Song et al. 2002). Team composition is especially critical in complex problem solving since it is a key enabler for integrating diverse sources of specialized knowledge as is required in dealing with complex problems (see § 2.2.1 and § 2.2.2). Specifically, in the development of complex products, the Integrated Product Team (IPT) has been the most commonly used team structure for over a decade (Browning 1996), due for the most part to the diverse team composition it provides. The cross-functional and multi-disciplinary composition of IPT's brings together multiple technical and functional knowledge domains for the entire lifecycle of the product, from concept design to testing and sometimes even sustainment (Sheard and Margolis 1995). This allows different specialists to tackle different aspects of complexity in the development task and to follow it through the entire development process (Ancona and Caldwell 1992; Browning 1996).

In large-scale development contexts, an IPT typically includes a variety of core specialists from the project or program developing the product, often supported directly on the team by domain experts from different functions in the prime organization, and sometimes even involving customer and supplier representatives, either as full-time team members or temporarily for specific phases of the development process. This cross-boundary composition of IPT's enhances the team's knowledge resources as well as its integration ability through enhanced coordination leverage and a larger organizational network to draw on. In such cases, an IPT's knowledge integration capability is largely dependent on the common language or knowledge base[22] established among its members. A common knowledge-base allows the members of a multi-disciplinary team to integrate their specialized knowledge internally by combining the different

[22] A common language or knowledge base that is shared by all team members is the basis for each member to be able to understand and assess the domain-specific knowledge of others (Carlile 2004), such as for example having common knowledge about "lifecycle design requirements" which addresses lifecycle design needs from the different perspectives of all engineering disciplines represented on the team.

types of knowledge they each possess, and in the process potentially creating new knowledge in the form of novel solutions or approaches[23] (Nonaka and Takeuchi 1995). In the absence of a common knowledge base, a team becomes a collection of disparate specialists with its members likely to be pulled in different directions and/or having to go through numerous iterations before reaching consensus on formulating and solving problems (Sheard and Margolis 1995). Therefore, a product team composed of diverse specialists with a common knowledge base between them and with the widest representation of the organizations involved in the development effort will be more efficient and effective at integrating knowledge both internally and from external sources.

Team leadership is arguably the most important determinant of team structure and has a direct influence on knowledge integration both internally and externally. As has been already shown in the discussion on problem solving teams in § 2.2.3, complex problems encountered in the development of modular systems are typically decomposable enough that they can be solved independently by different specialists under the supervision of a system architect as team leader, (Ulrich 1995) and the integration of knowledge would be mostly internal to the team or along vertical channels between teams at different levels of organizational or network hierarchy (e.g. prime and supplier organizations or subsystem and system IPT's in the same organization). Conversely, the development of integral systems requires joint problem solving by multi-disciplinary teams under the leadership of an experienced (or "heavyweight") system integrator, and the integration of knowledge requires extensive horizontal interactions between teams within the same organization. In addition, team leadership has a significant effect on the composition of the team since the team leader typically selects the individual members of the team, such that if the team is composed of individuals who are incompatible with each other in terms of their work styles, their interpersonal relationships or otherwise, the team is likely to be inefficient and/or ineffective at integrating knowledge due to conflict or broken communication links.

The team leader also plays a significant role in mediating conflicts internally and facilitating access to needed resources externally. This is why when team problem solving hits a "glitch",

[23] This process is often referred to as "learning-by-doing" and typically involves several iterations of trial and error as knowledge is shared, combined and applied to solve problems (Walz, Elam et al. 1993).

team leaders are typically called upon to put the problem solving back on track. Thus, the type of technical skills of the team leader as well as his or her leadership skills both affect the ability of the team at integrating knowledge resources internally and from external sources. It is important to note here that since the team leader acts as a funnel for knowledge flow into the team, team leaders must therefore be empowered by the organization to access and acquire knowledge from outside their local (program) boundaries in order for the team to effectively integrate knowledge from external sources that are not in their immediate vicinity.

Similarly, team size is an important aspect of team structure since it affects the ability of the team at integrating knowledge internally. The most common team size for IPT's in complex product development is typically 10-15 individuals (Browning 1997). If the size of the team is too small (less than 5 members) or too large (greater than 15 members), the team will likely be inefficient at integrating the knowledge required to accomplish its task within time and budget constraints (Browning 1997). This is because smaller IPT's may not have all the resident knowledge required in a complex development setting. Conversely, and while team size can sometimes be in excess of 30 members especially in large scale development contexts as is common in the aerospace industry, such teams are considered too large for members to work together efficiently and effectively. Group dynamics dictate that such teams eventually break up into smaller groups focused on specific tasks within their focus of expertise.

It should be noted here that even when team composition and size are close to optimal, it is often difficult for team members to embody all the knowledge required for a particular project, especially for large-scale complex development efforts, so the team will more often than not be forced to acquire additional knowledge from outside its own boundaries in order to solve complex problems. The sources of this knowledge can be relevant documentation, formal training sessions, or group meetings with other teams, among others (Walz, Elam et al. 1993; De Boer, Van Den Bosch et al. 1999).

Finally, the level in the organizational hierarchy at which an IPT is designed to operate dictates the types of channels and mechanisms that are available to its members for integrating knowledge internally and from outside of the team's boundaries (Galbraith 1974). In product

development, IPT's are structured to mirror the Work Breakdown Structure (WBS) of the development process, with different IPT's assigned to develop different parts of the product. These IPT's are known as subsystem IPT's, while the IPT responsible for the entire product is known as a system IPT (Browning 1996; Browning 1997). IPT's at all levels integrate knowledge between them through the exchange of design and interface documents, through intra- and inter-IPT meetings, co-location of individuals from different organizations or from different levels in the same organization, and common IT tools and databases (Browning 1997). However, system IPT's typically have a higher level of authority than subsystem IPT's in terms of determining and approving the problem solving approach, while the latter are more engaged in the actual steps and details of the problem solving process.

The key determinants and critical characteristics of team structure in large-scale complex product development are summarized in the matrix below:

Table 17: Key Determinants of Team Structure in Complex Product Development

Key Determinant	Critical Characteristics
Team composition	IPT format (multi-disciplinary, cross-organizational) Team member skills (communication, social networking, concurrent engineering)
Team leadership	Team leader skills (professional networking, mediation, system architecting, system integration)
Team size	$5 < \text{size} < 20$ Problem solving team meetings (10-15 members)
Team authority level	System-level IPT (oversight, approval of problem solving approach) Subsystem level IPT (implementation of problem solving steps)

To sum up, the primary vehicle for integrating knowledge in a complex problem solving context is the multi-disciplinary team, with IPT's being the most typical form used in complex product development. IPT's operate at various levels of the organizational hierarchy, and the structure connecting different teams mirrors that of the product architecture, which makes knowledge integration between teams a multi-level process. The ability of an IPT at integrating knowledge depends on the team's internal structure, with team leadership being arguably one of the most

critical aspects for both internal and external knowledge integration, as these depend on both the technical and leadership skills of the team leader.

2.3.7 The Link of Knowledge Integration to Technology Maturity

It has long been established in both the knowledge management and product development literatures that technological innovation has a direct effect on communication and coordination as well as on the structure of teams and organizations involved in the development effort. Earlier Work by (Allen 1997) demonstrated that the need for communication and coordination diminishes with increasing technological maturity as existing knowledge becomes stable enough that there is less need for engineers to stay connected with knowledge sources outside their team or project. As such, problem solving and knowledge integration in mature technology environments are typically localized and internal to the project team, whereas in novel or rapidly changing technology environments there is an increasing need to integrate knowledge across team and functional boundaries (Allen 2000).

Similarly, (Hansen, Nohria et al. 1999) argue that mature product development involving well-understood technologies and tasks can benefit from knowledge codification and reuse, whereas product innovation is best supported by tacit knowledge sharing among individuals since codified information becomes obsolete with rapidly changing technologies. All of these existing findings can be easily related and visualized in the following 2x2 matrix shown in Table 18:

Table 18: Effects of Technology Maturity on Complex Product Development

New Technology Environment	Mature Technology Environment
Distributed problem solving	Localized problem solving
Inter-team and program-to-function knowledge integration	Intra-team and intra-program knowledge integration

In complex product development, increasing technology novelty adds to the complexity of technical problems such as coupling problems resulting from incidental (or undesirable)

interactions between subsystem or component elements of a complex product, making it more difficult to anticipate such problems and catalyzing the formation of ad-hoc problem solving teams with experience in system integration, or even formal troubleshooting teams for particularly complex coupling problems (Eppinger and Gulati 1996). In a large-scale development context, technology novelty drives an increase in tacit knowledge interactions across prime-supplier boundaries, specifically to coordinate the integration of the supplier's subsystem module into the overall system (Brusoni, Prencipe et al. 2001). In contrast, when the product's core technology matures the relationship between prime and suppliers becomes more loosely coupled or at "arm's length", with interactions limited to explicit knowledge about subsystem requirements, specifications and standards. (De Boer, Van Den Bosch et al. 1999) further demonstrated that firms operating in a mature technological environment require systems capabilities (i.e. an information technology infrastructure) in order to integrate existing technological knowledge internally, while emerging technologies require the integration of new knowledge across organizational boundaries through group coordination which they define as participation in group problem solving, supplemented by liaison devices and lateral job rotation internally. Combining the preceding insights, the above matrix can be further refined as follows in Table 19:

Table 19: Effects of Technology Maturity on Complex Product Development

New Technology Environment	Mature Technology Environment
Integrated problem solving with supplier	Arm's length problem solving with supplier
Tacit knowledge integration using group coordination, liaison devices and job rotation	Explicit knowledge integration using IT infrastructure

In summary, the use of new and unproven technologies (or new combinations of existing technologies) increases the need for tacit knowledge integration vertically through tightly integrated group problem solving and troubleshooting with suppliers, and horizontally through job rotation and liaison devices between programs and functions. In contrast, the use of mature technologies can be efficiently supported by the integration of codified knowledge through an IT infrastructure.

2.3.8 Summary of Insights on Organization Design and Complex PD

In this section I have reviewed the main issues related to the organizational and product characteristics of influence on the knowledge integration process. These are summarized in Table 20 below:

Table 20: Knowledge Integration by Product and Organizational Characteristics

Product Characteristics	Organizational Characteristics	Primary Knowledge Integration Characteristics		
		Where	What	How
Low Complexity	--	Intra-firm	Explicit (few internal sources)	Documents; Information systems
High Complexity	--	Intra- and Inter-firm	Tacit (many internal and external sources)	Team meetings; Site visits; Liaison; Social network
Modular Architecture	Loosely Coupled Structure	Inter-firm	Explicit (subsystem-level specs, std's, reqts)	Documents; Information systems; Prototypes
Integral Architecture	Tightly Coupled Structure	Intra-firm (intra-team)	Tacit (system-level design/integ. knowledge)	Intra-team meetings; Mediation
New Technology	--	Intra-network, Prog.-function	Tacit	Group coordination; Liaison devices; Job rotation
Mature Technology	Loosely Coupled Structures	Inter-firm (prime-supplier)	Explicit	Documents; Information systems
New Product Design	--	Intra-program, prog.-function	Tacit	Intra-team meetings; Personnel transfer
Platform Product Design	--	Program-prog., Prog.-function	Tacit and Explicit	Inter-team meetings; Liaison Boundary objects
	Tightly Coupled Structures	Intra- and Inter-firm	Tacit (system-level design/integ. knowledge)	Face-to-face meetings; Social network
	Loosely Coupled Structures	Intra-network	Explicit (subsystem-level design knowledge)	Documents, Information systems

2.4 Summary of Literature Review

This chapter provided an overview of the literature on knowledge integration, problem solving, and organization design and complex systems development as they relate to the actual process of integrating knowledge in a large-scale complex product development context. Most notably, definitions for knowledge integration at the conceptual and operational levels were proposed in § 2.1.6. Summaries of the main insights related to knowledge integration from each of the main bodies of literature were presented in § 2.1.12, § 2.2.5 and § 2.3.8 respectively. The major themes related to the mechanics of knowledge integration which were commonly found across the literature are consolidated in Table 21 below. These insights form the basis for proposing the preliminary framework for knowledge integration in the context of large-scale complex systems development as presented in Chapter 3 of this thesis.

Table 21: Common Themes for Knowledge Integration

Knowledge Integration Channels	Problem Solving Context	Knowledge Integration Mechanisms
Intra-firm	Routine	Documents, information systems, liaison devices, boundary objects, communities of practice
	Non-routine	Personal communication, group interaction, people transfer, personal networking, team meetings, special taskforces
Inter-firm	Routine	Documents, information systems, networks of practice
	Non-routine	Group interaction, site visits, co-location, mediators, team meetings, special taskforces

3. CONCEPTUAL FRAMEWORK

Building on the existing insights in the literature as reviewed in Chapter 2 of this thesis, a preliminary framework is proposed conceptualizing the main dimensions for knowledge integration and problem solving in large-scale complex product development environments. The proposed framework will serve as a "going-in" proposition for the field research that can guide the data collection process in terms of the choice of interview subjects and research questions, as detailed in Chapter 5 on the methods used in the field research. The framework will then be updated and refined further as the data are collected and analyzed, which is shown in detail in Chapter 6 on data analysis. The steps for developing the preliminary framework are presented in the following sections.

3.1 Basic Dimensions of Knowledge Integration

The main insights distilled from the literature for framing the knowledge integration process have already been summarized at the end of each section of the literature review in the previous chapter. In order to develop a framework for knowledge integration in a complex product development context, it is necessary to tie these insights together into a typology of basic elements or dimensions for the knowledge integration process, characterizing the types and sources of knowledge and the strategies, practices, channels, boundaries and mechanisms for integration. The main dimensions for knowledge integration are visualized in Figure 7 below.

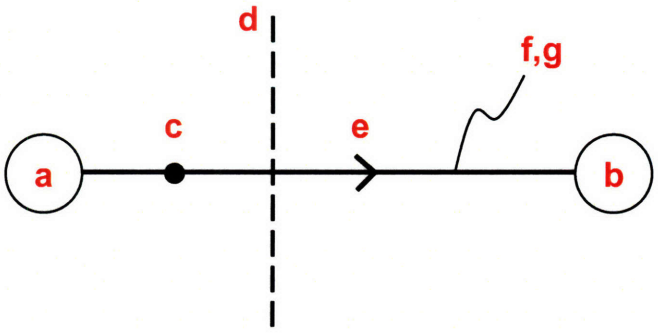


Figure 7: Main Dimensions for the Knowledge Integration Framework

Where:

[a] and [b] are counterparts involved in the knowledge integration process

[c] is the type of knowledge integrated

[d] is the type of organizational boundary being crossed

[e] is the direction of knowledge integration

[f] and [g] are the types of channels and corresponding strategies/practices/mechanisms employed to integrate knowledge along each channel

Note that the visual representation proposed above is fundamentally similar to a social network diagram composed of nodes tied by paths and separated by barriers[24]. This is because the focus in this research is on the actual process of integration as defined in the previous chapter, namely focusing on the transfer, sharing and application of knowledge between multiple sources and recipients across organizational boundaries, which makes network mapping a useful approach for visualizing the main knowledge interactions across the organizational network.

Using existing insights in the literature to characterize the above dimensions, a typology of basic "building blocks" for knowledge integration in complex product development can be proposed, as detailed in the following sections.

3.1.1 Typology of Knowledge in Complex Product Development

Developing a typology of knowledge serves two purposes of primary importance in this research, the first is to scope the research to include only those knowledge interactions which are related to complex problem solving during the product development process, in line with the original objective in this research; and the second is to actually identify and describe the main types and sources of knowledge that are most relevant in this context. First, and in order to scope the research so that it becomes operational, the investigation into the knowledge integration process will be focused on knowledge interactions occurring during the design phase of the product

[24] It is well established that "organizations can be viewed as social groupings" (Weick, 1969) and that "the social network perspective is an example of a theoretical framework...guiding data collection as well as data analysis...to capture significant organizational processes at different levels of analysis" (Tichy et. al, 1979).

development process, and specifically the focus will be on the integration of engineering knowledge during problem solving involving subsystem design and system integration type problems. This is because most technical problem solving occurs during the design phase of development (Moir and Seabridge 2004) where many teams are engaged in solving technical problems, as shown in Figure 8 below[25]. This was confirmed by interviewees in the field who stated that "the system design and development phase (SDD) is where you get the most interactions between the best and brightest engineers and with the most experienced experts". Secondly, the focus on technical problems occurring in these specific phases of development ensures comparability between similar problems of significant complexity, which is essential for framing the knowledge integration process in its complex setting. As one interviewee put it "the type of problems you encounter depends on the development phase you're in".

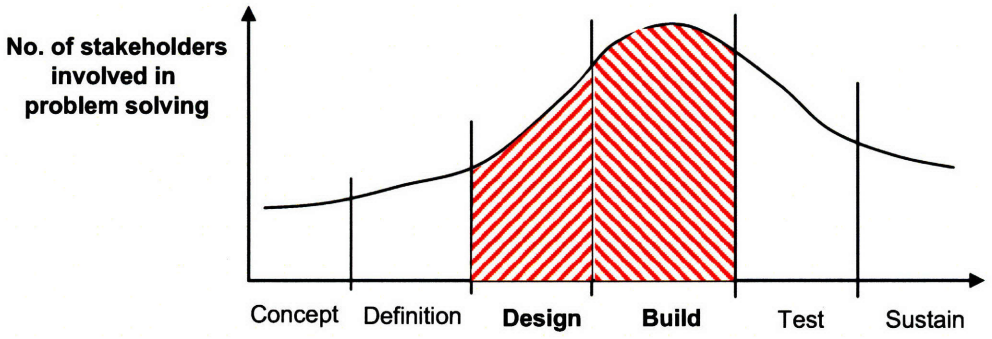


Figure 8: Research Focus on Design Phase of Product Development Process

It follows that knowledge of most relevance in this research is product-related knowledge about the technical aspects of the design and about the embedded technologies in the product, as well as the associated technical processes employed in the design phase of product development. Based on the insights in the literature as reviewed in the previous chapter (specifically in § 2.1.6 and § 2.3.2), a typology of such knowledge would consist mainly of component knowledge (meaning the engineering knowledge about the design of a particular subsystem and its

[25] Note that in many complex development projects, the different phases of development are not clearly delineated and some phases are frequently merged together due to schedule or budget pressures; as a result, the investigation of knowledge integration may include insights from the build and/or testing phases of development.

individual components) and architectural knowledge (or system integration knowledge about the interfacing of different components and subsystems), as well as process knowledge (or knowledge related to technical processes used in product development, such as materials engineering processes or software integration processes) (Henderson and Clark 1990). The sources of these types of knowledge in complex product development are therefore the prime (system integrator) and supplier (component developer) organizations, as previously discussed in § 2.3. The sources of system-level knowledge are assumed to be the program engineers in system and subsystem teams, as well as functional and support engineers and experts in functional groups. The sources of component knowledge are assumed to be the component engineers and experts in functional groups and the supplier engineers. It can therefore be concluded that the boundaries being negotiated in the process of integrating knowledge are team boundaries at different levels within the same program as well as across programs and functions, in addition to organizational boundaries between prime and supplier. Note that in order to keep the field research manageable, only the main participants in technical problem solving are included in the framework, thus excluding other sources of knowledge such as research and development teams, program management and the customer. A typology of the main channels for knowledge integration in large-scale complex product development can thus be proposed in Table 22 below:

Table 22: Typology of Knowledge Integration Types and Sources

Channel	Knowledge Types and Sources
Intra-program (system-to-subsystem teams) Channel #1	System integration knowledge from system architects, engineering leads and experts
Intra-program (subsystem-to-subsystem teams) Channel #2	System integration and subsystem engineering knowledge from systems engineers
Program-to-program Channel #3	Subsystem engineering knowledge from program engineers
Program-to-function Channel #4	System integration, subsystem engineering and process knowledge from functional and support engineers
Prime-to-supplier Channel #5	Subsystem engineering knowledge from supplier engineers

This typology frames the relevant types and sources of knowledge at different levels of integration (namely the subsystem and system levels, the prime and supplier levels, and the program and function levels) and along horizontal, vertical and lateral dimensions, which is in line with insights from the literature about the channels for knowledge integration in complex product development settings (see § 2.1.7 for a detailed discussion of the dimensions for knowledge integration). A typology of mechanisms for knowledge integration, constituting the remaining elements necessary for framing the process is discussed in the following section.

3.1.2 Typology of Integration Mechanisms in Large-Scale Product Development

Technical knowledge is interacted through tacit and explicit mechanisms, such as personal communication and group interaction for integrating expertise, know-how and skills, and documents and databases for integrating data and information, respectively. In addition, a fundamental assumption from the product development literature is that engineering knowledge at the component level can be effectively codified into explicit forms, such as documented specifications or standards, while system-level knowledge is more difficult to codify and is largely in the form of tacit expertise and experience held by individuals, such as expertise in systems engineering and system integration (Cusumano and Nobeoka 1998; Aoshima 2002). While this does not mean that subsystem-level knowledge is entirely codified (indeed design and engineering expertise with a particular subsystem is evolved by the supplier over several product generations and is closely protected from competitors) or that architectural knowledge cannot be captured in explicit form (for example, in system architecture drawings and charts or system-level specifications and standards), it is nonetheless expected that interactions at the component or subsystem level are dominated by explicit knowledge integration mechanisms while interactions at the system level are dominated by tacit knowledge integration mechanisms.

Based on existing insights in the knowledge integration and product development literature as reviewed in Chapter 2 of this thesis, and building on the classification in the previous section of the main types and sources of engineering knowledge in complex product development, a typology of the main integration mechanisms in this context is proposed in Table 23 below:

Table 23: Typology of Knowledge Integration Mechanisms

Channel	Knowledge Types	Integration Mechanisms
1. Intra-program (system-to-subsystem teams)	Information about system requirements	System architecture drawings and system requirements documents and databases
	System integration expertise	Team meetings; personal communication
2. Intra-program (subsystem-to-subsystem teams)	Information about subsystem requirements	Subsystem design drawings and interface control documents and databases
	System integration expertise	Team meetings; personal communication
3. Program-to-program	Subsystem engineering knowledge	Subsystem design documents and databases; communities of practice
4. Function-to-program	Information about system and subsystem specifications and standards	System subsystem design documents and databases; process documents and tools
	System integration and subsystem engineering expertise	Team meetings; liaison devices
5. Prime-to-supplier	Subsystem requirements, specifications and standards	Subsystem design drawings; subsystem requirements documents and databases; boundary objects
	Subsystem engineering expertise	Team meetings; liaison devices; networks of practice

The proposed typology consists of the most dominant mechanisms used to integrate knowledge during product development across the five main channels, as specified previously. The mechanisms and practices shown above map to typical product development tasks which include determining customer needs (i.e. customer requirements), converting those needs to engineering and process specifications and ultimately to a final product (Fine and Whitney 1996).

3.2 Towards a Framework for Knowledge Integration

Having specified the main types and sources of knowledge and the corresponding integration

channels and mechanisms, a simple framework is proposed to visualize the main knowledge interactions in a complex product development context, as shown in Figure 9 below:

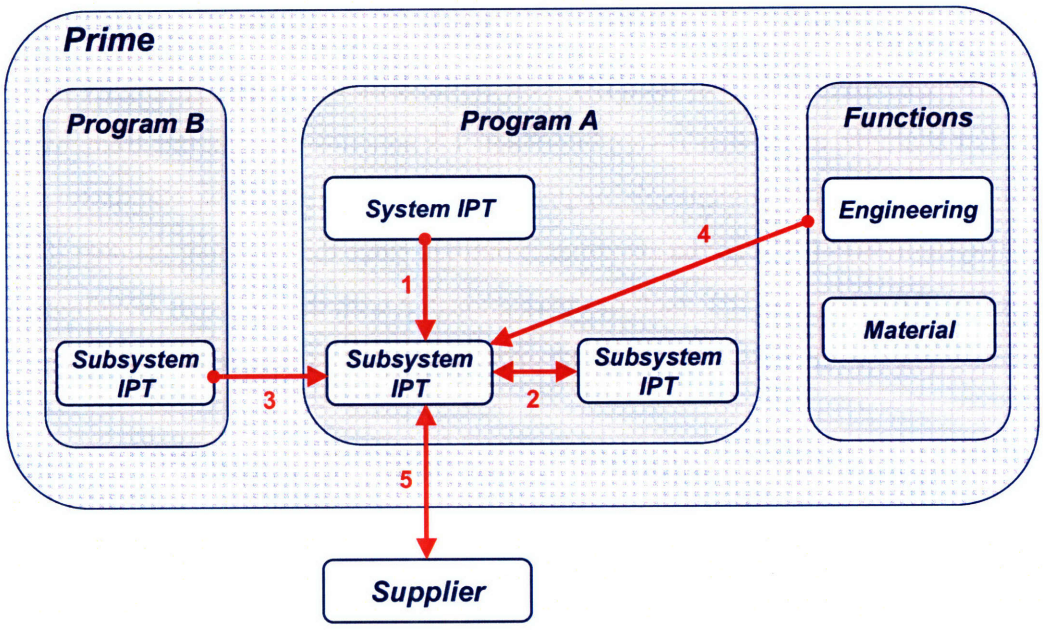


Figure 9: Proposed Framework for Knowledge Integration

The framework above depicts the main counterparts involved in the knowledge integration process in a large-scale organizational network and the major channels for integrating knowledge from the perspective of the subsystem IPT in "Program A", which is considered as the locus of problem solving in complex product development (i.e. where the problem solving process starts and where the majority of problem solving occurs). This node is shown at the center of the framework since it is the main beneficiary in the knowledge integration process. The proposed structure constitutes a multi-level framework for knowledge integration where vertical channels map to the main levels of the product's design hierarchy (i.e. from component and subsystem level to the overall system level).

The subsystem IPT is therefore the unit of analysis in this framework since it is a major source of knowledge as well as the main recipient and target in the integration process. The links between the nodes in the depicted structure are a high-level abstraction of the main integration channels

within and across organizational boundaries. The arrows in the figure illustrate the main direction of knowledge integration along each channel. These channels are meant to be representative but not exhaustive of the major knowledge interactions in a large-scale complex product development context, so that those channels which are not considered as critical for technical problem solving by subsystem IPT's are omitted for clarity and simplicity in the analysis. For example, a supplier-to-supplier channel is not considered in this framework since it is generally external to the problem solving process by the subsystem IPT. Similarly, a channel linking subsystem level teams to the customer is omitted since the customer is either directly represented on the subsystem IPT itself or the relationship with the customer is indirectly mediated by higher level teams or leads in the prime organization hierarchy.

Along those lines, there was no inclusion of channels linking the subsystem IPT to other program sources of knowledge such as the program management or the chief engineer groups, as these are considered synonymous with the system-level team for simplicity. Finally, channels to other corporate sources of knowledge such as research and development teams are also omitted since these groups are not frequently involved in problem solving. These simplifications are in the same spirit as the original strategy of bounding the research in order to keep it manageable. However, the framework remains comprehensive enough in terms of accounting for the parties involved in knowledge integration and problem solving alongside the subsystem IPT. Also, note that the adopted structure is typical of project-based organizations which are most common in complex systems development (Dosi, Hobday et al. 2000), therefore confirming that the main channels for knowledge integration in this context are accounted for. The proposed framework will serve as a "going-in" proposition in order to guide the field research into the actual knowledge integration channels, strategies/practices and mechanisms most frequently used in large-scale complex product development environments.

4. RESEARCH LENS

In order to bound the research problem and to ensure that the field investigation can be practically and efficiently accomplished, this research will focus on one particular type of complex engineering systems, namely military avionics (i.e. aviation electronics) systems, as a research lens into complex systems development. Specifically, the focus of the research will be on major "Mission Systems" commonly used in military aircraft and which closely interact together in order to deliver mission capability to the pilot. These are principally the Multi-Function (MF) Multimode Radar, the Electronic Warfare Suite (EW), the Communication Navigation Identification (CNI) system, and the Mission Computer (MC). Figure 10 illustrates how these systems are typically and approximately distributed on modern fighter aircraft and the types of functions they perform.

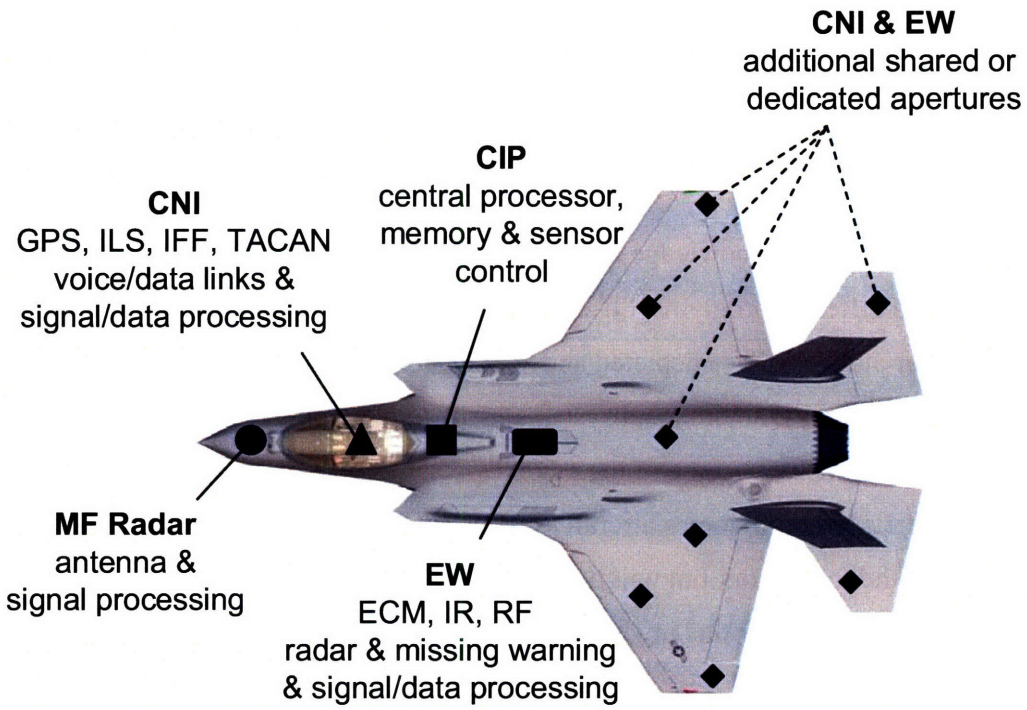


Figure 10: Typical Mission Systems, Apertures and their Functions

The generic framework for knowledge integration proposed in the previous chapter can thus be reframed for the context of avionics development, as shown in Figure 11 below:

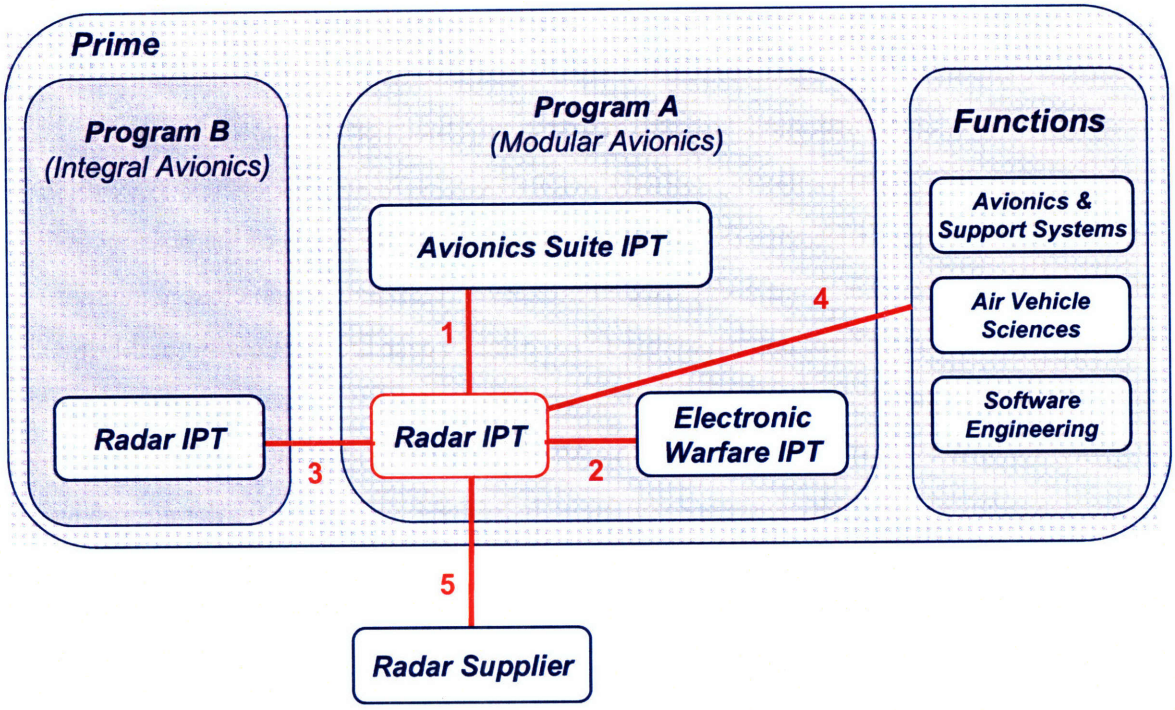


Figure 11: Knowledge Integration in Military Avionics Development

As the above figure shows, knowledge integration in this context would be along horizontal channels between different programs, different mission subsystem IPT's in the same program such as the Radar and EW IPT's depicted above, and along vertical channels to the suppliers of those subsystems as well as to other subsystems in the total avionics suite. The lateral channel illustrates the interaction with some of the avionics-centric functional groups. Note that for simplicity, channels to non-avionics aircraft systems such as mechanical and airframe systems are not shown in the framework but will be investigated in this research. The main reasons for choosing military avionics for this investigation are discussed in the following section.

4.1 Why Military Avionics

The rationale for choosing military avionics as a lens for the inquiry in this research is threefold: first, avionics systems are some of the most complex equipment currently flying on military aircraft, in terms of the number and variety of component parts in their makeup as well as the

interdependence between those components, both internally to each avionics subsystem and externally with other subsystems (Moir and Seabridge 2006). These components include electronic sensors, processors, data buses, software, power sources, mechanical actuators, digital controls, audio equipment and video displays, among others (Loewy 1999). As such, avionics systems make for a rich research context from the perspective of knowledge integration since the development of these systems is rife with complex problem solving activity in large-scale organizational settings, especially in terms of dealing with major technical problems having important strategic implications for competitive advantage, such as system integration problems, supply network integration issues, and technology and system architecture choices.

Secondly, avionics systems are highly knowledge intensive as they incorporate some of the most cutting edge technologies in the military aerospace and commercial industries, from micro-processing and fiber-optic networking to digital communication and flight control technologies, among many others (Spitzer 2001). In addition, most avionics technologies are rapidly evolving with a majority of component technologies having very short technology refresh rates on the order of 2 to 3 years (Committee on Aging Avionics in Military Aircraft 2001), which makes avionics systems prone to major redesigns and thus a primary target for technical knowledge integration during the development process. This offers the opportunity to frame the knowledge integration phenomenon under different technology maturity levels as well as for a variety of system architecture regimes which have emerged as a result of this fast clockspeed evolution (see § 4.2.1 for an overview of avionics technology and architecture evolution)

Finally, the choice of the defense context for this research offers a fertile ground for the investigation of knowledge integration, first in terms of the numerous restrictions over the exchange of classified information across organizational boundaries, such as the policy barriers against knowledge sharing with foreign suppliers (namely the International Traffic in Arms Regulations (ITAR)), and second, due to the largely traditional mentality in military development where open knowledge sharing is not widely adopted. Furthermore, military avionics systems are highly customized and there is no established tradition for commonality and knowledge sharing across platforms in this context. These and other challenges in the development of military systems offer the potential for significant insights about effective

enablers and workarounds for the integration of knowledge across impermeable organizational boundaries. The following section will provide an overview of the evolution of military avionics systems and a summary of the corresponding technical and organizational issues of relevance for this research.

4.2 Avionics Overview

The primary role of military avionics systems is to provide mission capability to the pilot, from target identification and weapons aiming in air superiority missions to intelligence gathering and precision bombing in surveillance and ground attack missions, among others. The ever-increasing need for superior mission capability has led to continuous evolution and growth in avionics system capabilities, as well as in the cost of avionics development and lifecycle maintenance. However, the evolution of military avionics systems over the past two decades reflects the changing imperatives in the defense aerospace industry, where affordability in the post-cold war era became the primary system requirement due to shrinking defense budgets. And with avionics systems now making up on average an almost 30 to 40 % of the total development cost in new military aircraft (Joint Advanced Strike Technology Program 1994), and over 30% of the US Department of Defense's (DOD) depot maintenance costs (Committee on Aging Avionics in Military Aircraft 2001), it is easy to see how affordability and system capability in avionics can be enhanced through the efficient and effective integration of new and existing knowledge, including engineering and process knowledge about technologies and system architecture choices, and design and system integration knowledge.

Like other complex products, avionics development starts with customer requirements about the critical functions to be fulfilled by the avionics system, as well as about performance targets and constraints, costs, and schedule information. These requirements are structured and formalized before being flowed down to subsystem level IPT's and suppliers in charge of developing different avionics subsystems. During the development process, the developing IPT's also interact with functional groups and other programs to integrate engineering knowledge about various aspects of the design and integration of their particular subsystem. Building on the typology of knowledge and knowledge integration channels proposed in § 3.1.1, the following

Table 24 gives an overview of the main engineering skills required and the types of knowledge interacted in military avionics development (Spitzer 1987; Moir and Seabridge 2001):

Table 24: Typology of Knowledge Integration Types and Sources

Channel	Knowledge Types and Sources
Intra-program (system-to-subsystem teams) Channel #1	Functional and performance requirements, interface specifications and standards and other system integration knowledge at aircraft system level
Intra-program (subsystem-to-subsystem teams) Channel #2	Interconnect technologies, specifications and standards, sensor fusion, resource management and other avionics subsystem integration knowledge
Program-to-program Channel #3	Avionics design knowledge (signal processing, packaging, signatures, apertures)
Program-to-function Channel #4	Avionics technologies (microcircuit, RF components, artificial intelligence) and avionics design and integration knowledge
Prime-to-supplier Channel #5	Functional and performance requirements, interface specifications and standards, microcircuit technology (memory, processors) and other avionics subsystem design knowledge

The research will use the typology above to investigate the frequency of integrating particular types of knowledge along the corresponding channels and the types of strategies and mechanisms used in the integration process. Furthermore, another of the main objectives in this research is to determine the influence of different system characteristics on the process of integrating knowledge, with one of those main characteristics being the type of architecture for the system under development. The following section will provide an overview of the evolution of avionics systems and the dominant system architecture regimes in modern avionics.

4.2.1 Evolution of Avionics Architectures

Figure 12 shows a typical system architecture for modern military avionics suites:

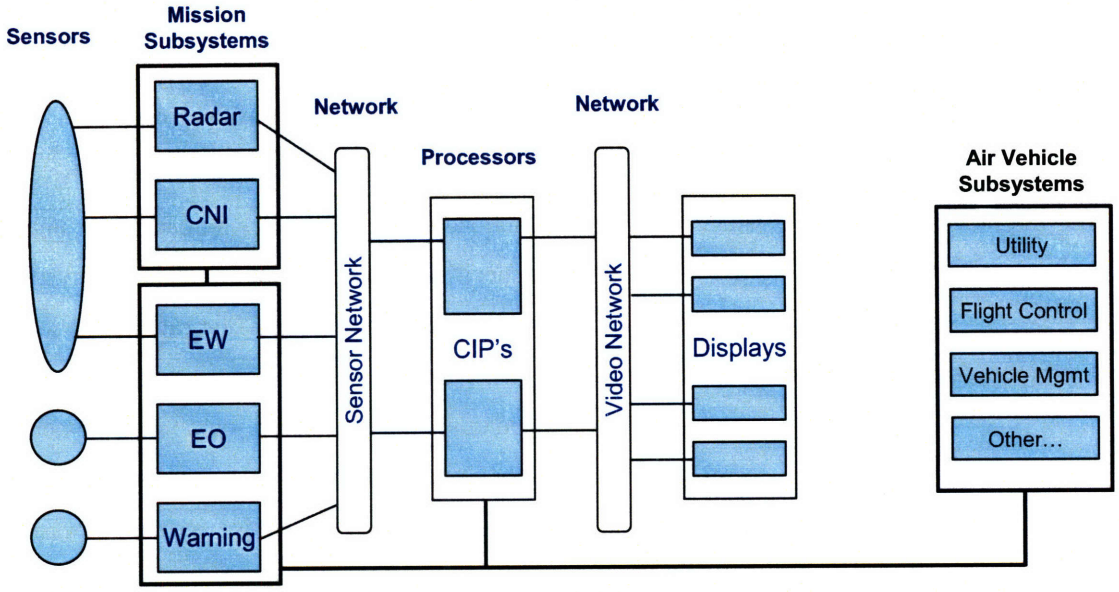


Figure 12: Typical System Architecture for Advanced Military Avionics

During the cold war, the defense sector was still leading the commercial world in avionics technology development, and avionics systems architectures (i.e. the physical, functional and logical configuration of avionics systems) followed a highly integrated design as an effective means of delivering custom performance requirements at any cost. However, in the post cold-war era, a shift towards more modular architectures began to develop, driven by the imperative for affordable systems and the opportunity to leverage the vastly increased performance and reduced cost of computer processing and memory modules from the commercial world. Since then, cyclic trends of back-and-forth integration and modularization have ensued, leading to the emergence of different system architecture regimes across aircraft generations. This makes avionics development especially interesting from the perspective of knowledge integration as these systems offer a wide variety of system architectures, which are briefly reviewed here.

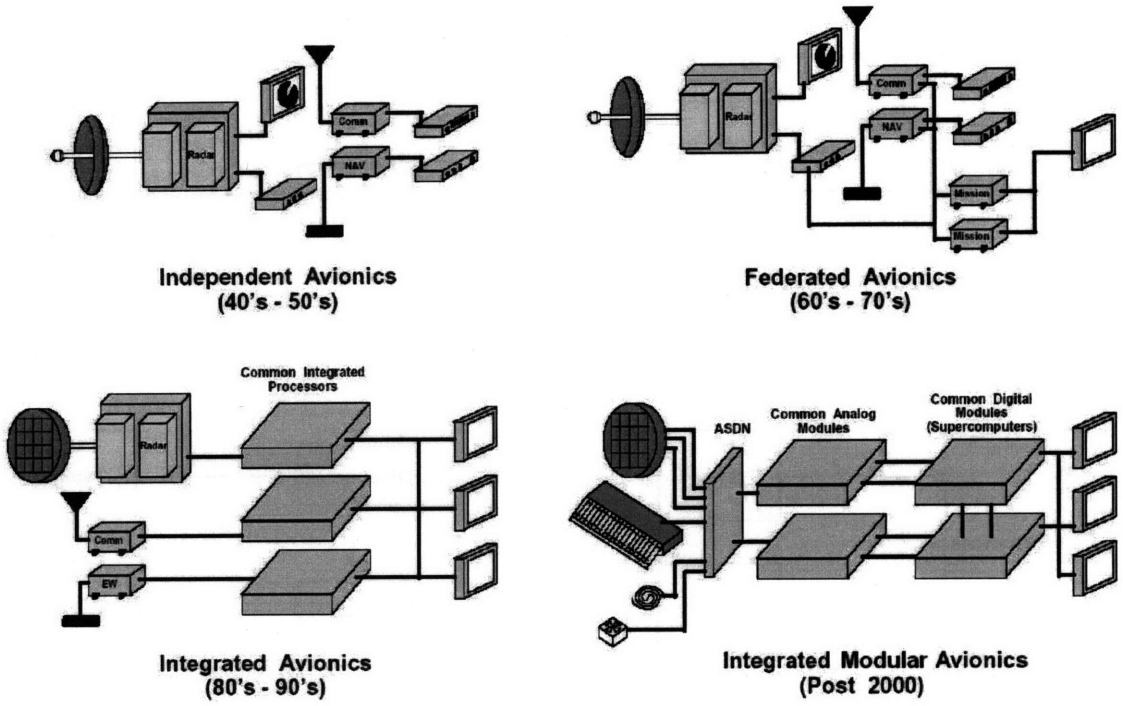
The first generation of avionics systems consisted of separate and independent systems each dedicated to perform a single function, and each hard-wired to its own dedicated sensor, analog processor and display, leading to numerous bulky systems and forcing the aircrew to be the integrators of the information output (Quaranta 2000; Filmer 2003). The advent of the digital

computer and its ability to process information from multiple sensors led to the second generation of avionics architectures known as federated avionics (due to the single processor feature for many separate and dedicated functions). Federated avionics are otherwise integrated parallel processing systems with mission computers controlling the different functions through parallel bus-structured interconnects (Filmer 2003; Hicks 2004). This architecture allowed more data and signal processing over its predecessor, with the information from different functions all presented to one display, one function at a time. It also provided significant weight savings due to reduced wiring interconnects between systems. However, the utilization of unique resources by each system led to higher initial as well as lifecycle costs; still, most military aircraft flying today are equipped with federated avionics architectures, with a large number of them undergoing modernization programs. Notable among these programs is the USAF Modular Avionics System Architecture (MASA) program started in the late 1980's, which sought to expand the use of modularity in the modernization of federated avionics. One of the more significant findings of this program was that modular architectures would enhance the reliability of avionics systems through increased redundancy and fault tolerance, which would boost the overall operational availability of the aircraft as well as reduce lifecycle maintenance costs (Brock and Schor 1990).

Also in the late 1980's, the PAVE PILLAR project was initiated with the aim of standardizing avionics architectures, starting with the Lockheed Martin F/A-22 Raptor for the USAF. The outcome was a third generation of architectures known as integrated avionics, which reflected a paradigm shift in architecture design from a loose interconnection of highly specialized "black boxes" to having a system of general-purpose computers known as common integrated processors (CIP) sharing many tasks and large amounts of data. Integration of previously separate functions and sharing of resources (e.g. memory sharing) delivered increased situational awareness to the pilot through improved sensor data fusion and provided significant cost and weight savings over federated architectures due to fewer required physical devices. However, the added architectural and functional complexity (i.e. the tight coupling of functions) led to significant cost growth and schedule delays during design and development. In addition, the interdependencies between hardware and software later led to increased costs of modernization, since any changes in one domain forces upgrading in the other at the same time. And despite the

initial intent of the PAVE PILLAR project in creating an open flexible architecture, integrated avionics were essentially closed (i.e. proprietary) architectures with limited potential for growth (Morgan 1995; Quaranta 2000; Moir and Seabridge 2006).

The PAVE PACE project adopted lessons learned from PAVE PILLAR and was the basis for the fourth generation of avionics systems known as advanced integrated avionics or more commonly as integrated modular avionics (IMA), initially proposed for the F-35 Joint Strike Fighter (JSF) – now known as the F-35 Lightning II – a multi-role aircraft with several variations for different military customers. This latest generation of avionics is characterized by an open system architecture (OSA) with extensive use of commercial-off-the-shelf (COTS) hardware and software, and an increased focus on modularity at the overall system level. The key objectives of the OSA approach are to enable sharing of the same computer resources through processor and software commonality, and to decouple the software from the underlying bus and hardware architectures. This is accomplished by utilizing well defined commercial interface standards and using modular design concepts. This will lead to an architecture that is more flexible (by decoupling the processor module from software code), fault tolerant (through redundancy of component modules), scalable (easily upgradeable with improved components), interoperable and reusable (across platforms), which makes it easier and more cost-effective to maintain (Fabian and Rayl 1998; Filmer 2003). Figure 13 below illustrates the evolution of avionics system architectures over time.



Source: JAST, Avionics Architecture Definition, 1994

Figure 13: Avionics Systems Architecture Evolution

A more simple visualization of the different generations of avionics systems is shown in Figure 14 below, based on the most relevant characteristics that differentiate integrated from non-integrated systems, namely the connections between different systems (or subsystems) and their corresponding resources.

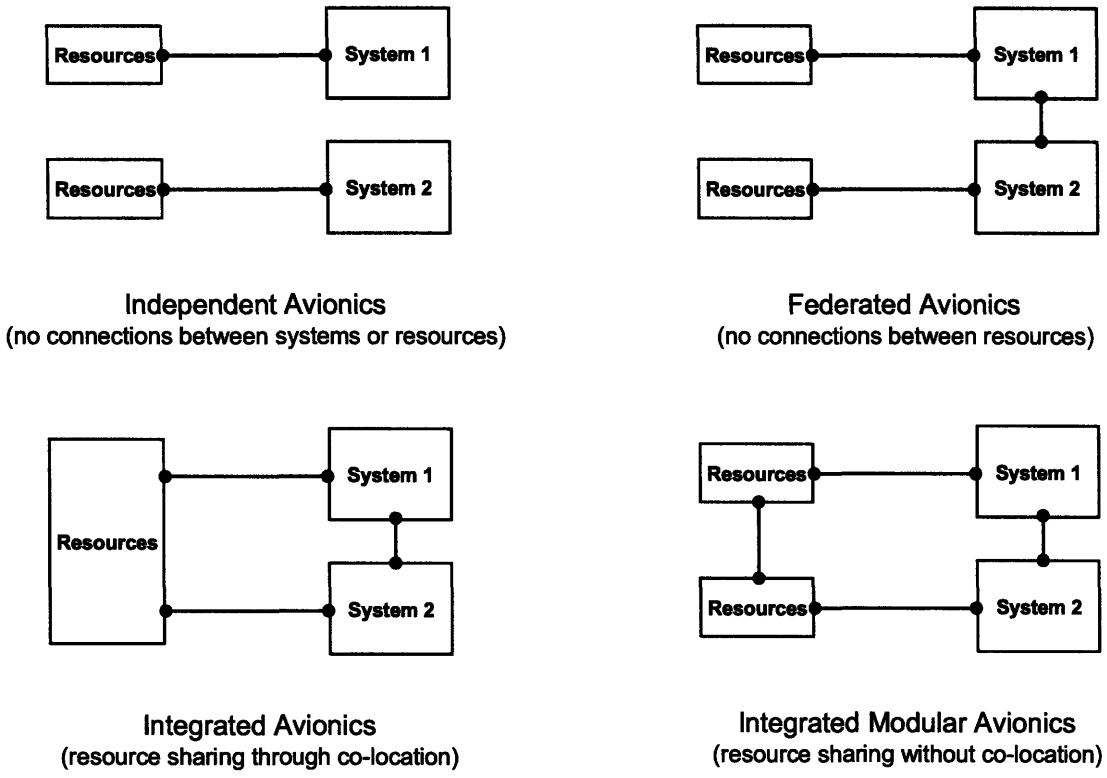


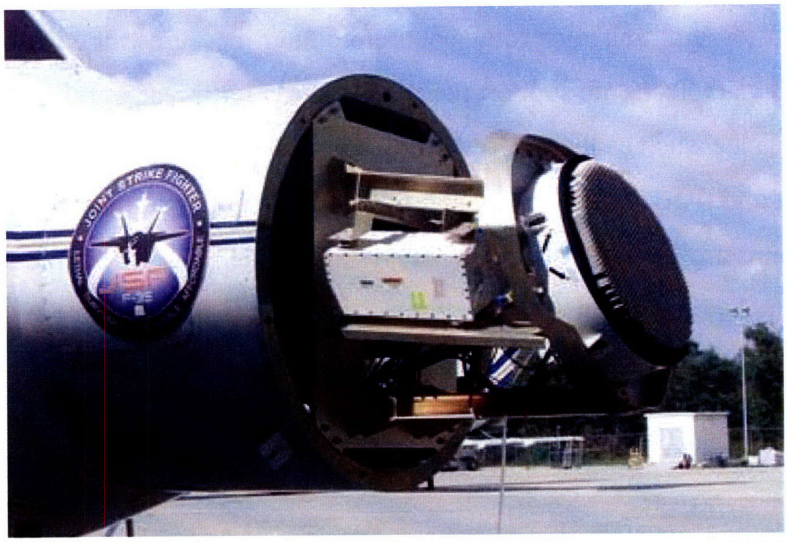
Figure 14: Main Characteristics of Avionics Systems Architectures

Following is an overview of the main functions and characteristics of the major mission avionics systems of interest in this research.

4.2.2 Overview of the Multi-Function Radar System

The basic principles of radar (which stands for Radio Aid to Detection and Ranging) are to transmit radio frequencies (RF), either continuously or in pulses, in order to detect the presence of other objects and to measure the distance to those objects using the time it takes for transmitted waves to be reflected back to the radar receiver. Airborne radar is thus used to detect the presence and range of other aircraft, and fighter aircraft are typically fitted with multimode radars capable of operating in different modes, such as air-to-air search, tracking, track-while-scan (TWS) and ground mapping modes (Moir and Seabridge 2006). Advanced military aircraft are equipped with highly sophisticated radar systems, such as the active electronically steered array (AESA) radar shown in Figure 15 which is capable of performing a number of

sophisticated functions simultaneously, such as synthetic aperture (SA) terrain mapping for air-to-ground surveillance, targeting and jamming functions.



Source www.northropgrumman.com, 2008

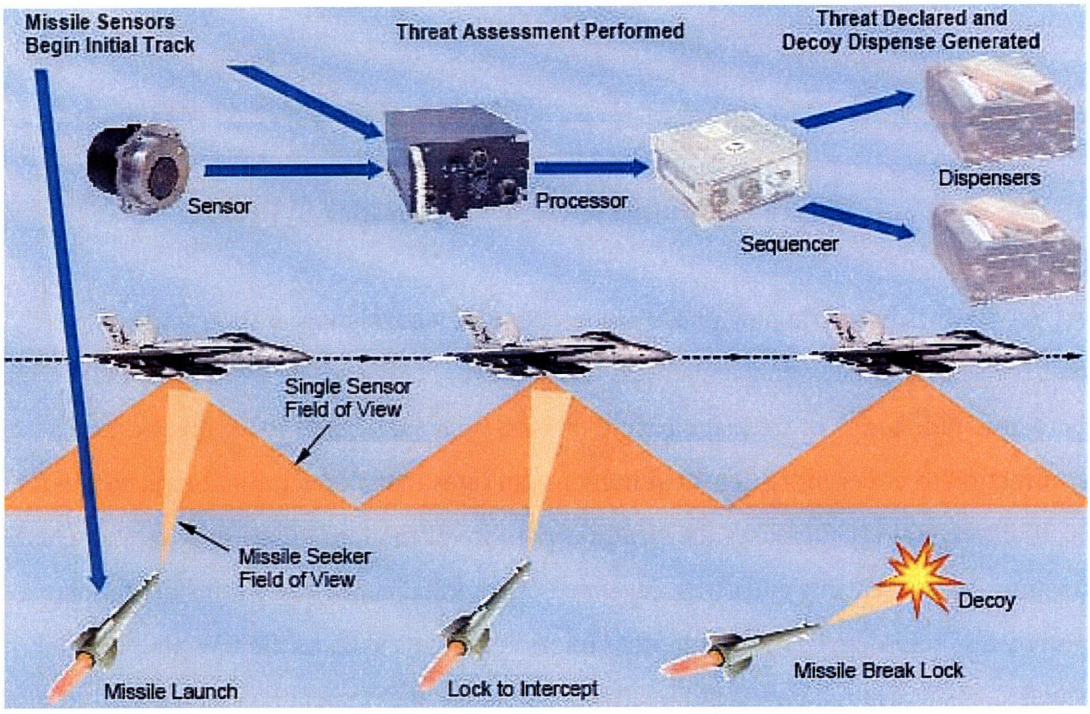
Figure 15: Multi-Function AESA Radar System

The AESA radar is based on new phase array technology consisting of thousands of transmit/receive (T/R) elements embedded in the radar antenna which are able to radiate multiple beams over a wide range of frequencies and phases, with each beam being electronically steered in any direction to cover more terrain at higher scan rates than conventional radars (Quaranta 2002). This technology increases the stealth capability of modern radars making them less prone to detection. However, this and other advanced operational capabilities translate to increased complexity and interdependence with other mission systems such as the EW and the CNI subsystems. As a result, a new trend in advanced radar development has seen a shift to more standalone systems with their own dedicated resources (i.e. more modularized from the rest of the avionics suite) in order to manage development complexity (Quaranta 2000).

4.2.3 Overview of the Electronic Warfare (EW) System

Electronic warfare (EW) systems serve to gather information and evade detection as well as to jam enemy radars and guided weapons. Modern airborne EW systems are equipped with

multiple sensors and receivers to cover emitting systems on the ground and in the air. Advanced EW systems consist of signals intelligence (SIGINT) equipment for collecting electronic intelligence (ELINT) and communications intelligence (COMINT), electronic counter measures (ECM) equipment for jamming, electronic counter-counter measures for countering enemy ECM and ESM systems, and electronic support measures (ESM) for threat warning, target acquisition and weapons guidance, such as the radar warning receiver (RWR), missile warning system (MWS) and chaff and flare dispensers, among others (Moir and Seabridge 2006). As such, modern EW systems are made up of several subsystems which closely interact together, making for highly integrated system architectures. Figure 16 below shows typical EW modules and their functions in a missile engagement.



Source www.baesystems.com, 2008

Figure 16: Simplified Overview of EW Subsystems on Fighter Aircraft

As the figure shows, when electromagnetic transmissions are detected by the EW system, it processes them to isolate each signal and identify it relative to a database of threats in order to determine if the source is a friendly or hostile transmitter, before taking further action such as

automatic jamming and decoying of enemy aircraft (Chaltiel, Gourion et al. 1998). EW systems also interact with other mission systems such as radar and navigation systems in order to locate the source of transmission relative to the aircraft. In some avionics architectures, EW and CNI systems share the same antennas which can lead to design dependencies between the two systems.

4.2.4 Overview of the Communication, Navigation and Identification (CNI) System

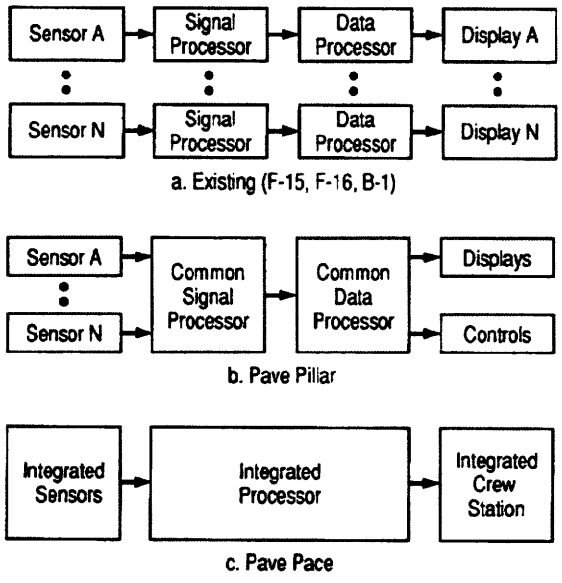
The communications, navigation and identification (CNI) system provides the aircraft with the ability to communicate by voice or data, to navigate to a target or waypoint, and to identify and classify targets before aerial engagement, respectively. The CNI system is made up of several equipment and sensors, including ultrahigh frequency (UHF) and very high frequency (VHF) communications, satellite communications (SATCOM), tactical air navigation (TACAN), traffic collision and avoidance system (TCAS), global positioning system (GPS), instrument landing system (ILS), and identification friend or foe (IFF). These equipment are typically separate but highly integrated modules and which interact together considerably in order to deliver superior functionality through data fusion, which makes the overall architecture of modern CNI systems closer to integrated modular architectures (IMA) (Wolfe, Campbell et al. 1996). Figure 17 below shows a modern CNI system for fighter aircraft.

Source DoD 1998

Figure 17: Advanced CNI System for Modern Fighter Aircraft

4.2.5 Overview of the Mission Computer (MC) System

The mission computer (MC) is the 'brain' of the entire avionics system. This system is responsible for data and signal processing from sensors and avionics missions subsystems, including the radar, EW and CNI. As such, the modern military mission computer is tightly integrated with other parts of the avionics suite (e.g. its components are physically distributed throughout the avionics system), while being a standalone system at the same time in terms of having its own resources such as power, cooling and memory to increase reliability. However, the internal architecture of the mission computer is highly integrated, with modern computers containing dozens of microprocessors for greatly enhanced capability. The system architecture trend for modern mission computers is therefore closest to the integrated modular regime (Imbesi and Kaplow 1992). The most advanced mission computers in development and operation today are the common integrated processor (CIP) for the F-22 Raptor, the integrated core processor for the F-35 Lightning II, and the modular mission computer for the F-16 Block 60 avionics upgrade. Figure 18 shows the architecture trend for military mission computer systems:



Source Robinson and Trujillo 1992

Figure 18: Mission Computer Architectures for Modern Fighter Aircraft

The choice of these specific subsystems which are common elements in the mission systems suites of all modern fighter aircraft makes it possible to have a more representative sample of major technical problems of interest and which are comparable for the purposes of this research, as discussed in the following Chapter 5 on methods for field research.

5. METHODS

This chapter is a review of the research procedures and methods employed to collect and analyze the data in both the interview part and the questionnaire survey part of the research. The following Figure 19 situates the field research in the overall thesis roadmap.

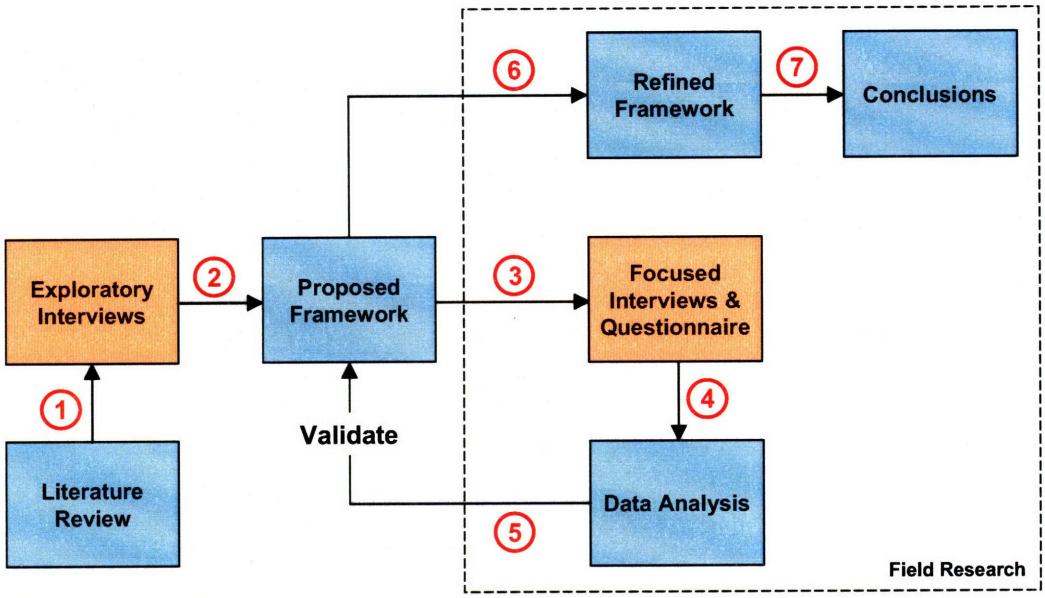


Figure 19: Research Roadmap

First, an overview of the research cases is provided in terms of the number of participating organizations and programs and the relationships between them during the avionics development process, as detailed in the following section.

5.1 Overview of the Research Cases

The field research was conducted with three military aircraft programs (herein referred to as Program "A", Program "B" and Program "C"[26]) common to three major defense contractors in the US defense aerospace industry (referred to as the Prime, Supplier #1 and Supplier #2). The

[26] In order to protect individual and company confidentiality, individuals, programs and organizations participating in this research will not be identified by name or any other identifying characteristics anywhere in this thesis. In addition, no information deemed to be proprietary or classified is used anywhere in this thesis (see Appendix "D")

supplier organizations are both major partners with the prime on the same three aircraft programs, whereby each supplier is tasked with developing the same avionics systems (or subsystems) across two or more programs. This selection strategy affords a "triangulation" of the collected data which ensures its cross-corroboration in accordance with commonly accepted practices for enhancing research quality (Robson 2002). This approach was also to ensure that substantial and substantive insights (i.e. rich in terms of quantity and quality, respectively) could be collected about knowledge integration across external organizational boundaries between the prime and its major partners, with the chosen suppliers being characterized as the prime's "most proactive partners" in the avionics development process (according to several interviewees in the prime organization). Figure 20 illustrates the relationships between the three organizations across all three programs. For simplicity, only one connection is shown from each program to each supplier organization; however the same multi-program structure is adopted at the suppliers as it is show in the prime organization.

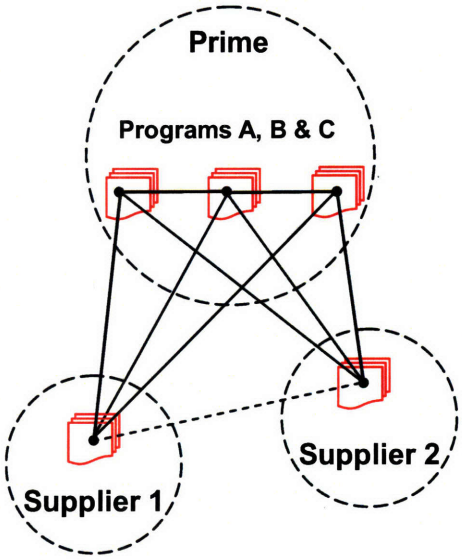


Figure 20: Overview of Research Cases and Interrelationships

The selection of the chosen programs was based on the variety of the total avionics architecture they provide, such that the total avionics suite in program "A" is considered to be an overall modular architecture, while program "B" is considered to have an overall integrated total system architecture and program "C" an overall integrated-modular architecture. Note that this does not

mean that all subsystems in a particular avionics suite follow the overall system architecture (e.g. this does not mean that all mission subsystem in program "A" are modular), only that the majority of subsystems tend to follow the overall trend in each avionics suite (e.g. there is more modularization between different subsystems in program "A", and more integration between the same subsystems in program "B", as a general trend). Table 25 below gives an overview of the relationships between the three organizations in terms of their individual and collective responsibilities in the development of the avionics mission systems of interest in this research for each of the chosen programs:

Table 25: Avionics Development Responsibilities of the Participating Organizations

Organizations / Programs	Program A	Program B	Program C
Prime	MC, CNI[27]	MC, CNI	MC
Supplier #1	Radar, EW	Radar	Radar
Supplier #2	--	EW	EW, CNI

The field research included two phases of data collection, a first phase which consisted of multiple rounds of exploratory and semi-structured interviews with individuals and teams across all three organizations, and a second phase where a structured questionnaire was administered to the same or different individuals than those who were previously interviewed. The vast majority of interviews were conducted on-site, with a few others through teleconference and over videoconference. All research was conducted in person and no data was collected online or by mail or email. An overview of the different stages of the field research is shown in Table 26.

Table 26: Avionics Development Responsibilities of the Participating Organizations

Participating Organizations	Interviews	Questionnaire	Total Rounds of Research
Prime	3 site visits (3 days each)	1 site visit (4 days)	4 site visits
Supplier #1	2 site visits (1 day each)	1 site visit (1 day)	3 site visits
Supplier #2	2 teleconferences	2 teleconferences	4 teleconference rounds

[27] The abbreviations "MC", "CNI" and "EW" are used to denote the "Mission Computer" system, the "Communication Navigation and Identification" system and the "Electronic Warfare" system, respectively.

In total, four rounds of on-site research were conducted with the prime organization including two videoconferences and four teleconference calls with other offsite locations. Also, three rounds of on-site research were conducted with supplier #1 and four rounds of teleconference calls were conducted with supplier #2. A total of 63 interviews were carried out with 50 individuals spanning multiple teams and divisions across all three organizations and covering all three programs, for a total of 61 hours of interviews. Of the 63 interviews, 20 were follow-ups with the same individuals, which served to explore earlier insights in a more in-depth way. The average experience of all interview subjects in avionics development is over 20 years. The distribution of interviews among the three programs and functional groups is shown in Figure 21 below:

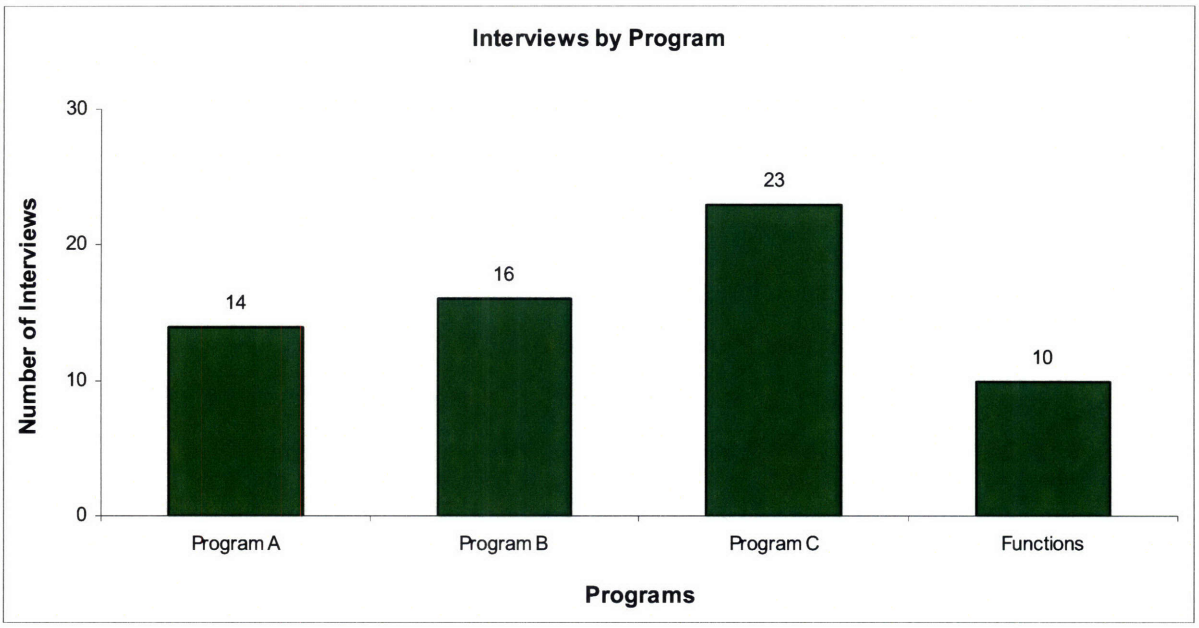


Figure 21: Interview Subjects by Organization (Total Interviewees = 50)

More interviews were conducted with program "C" than the other two programs and the functional groups since program "C" is still in the design phase while the other two programs have gone into production, and therefore more up-to-date insights could be collected about engineering interactions from program "C" than from the other two programs which are at more

mature phases of development (note however that the design and build phases in programs "B" and "C" are merged together and therefore there is still significant design activity currently going on at these two programs). Note also that more interviews were conducted with subjects from the prime than from the two supplier organizations in each of the programs surveyed. This was partly due to limited access to the supplier organizations, but was also driven by the larger scale of resources invested by the prime relative to its suppliers in each program. Therefore, despite the uneven distribution of interviews between the three organizations, the sample of interview subjects is in fact representative of the level of involvement of each organization in the overall integration of knowledge during development. Note also that the research method adopted for the collection and analysis of the interview data (namely the grounded theory method as detailed in § 5.2 below) does not require a random sample of interview subjects but is based instead on purposive sampling (known as *theoretical sampling* in grounded theory), the aim of which is to select those subjects who are closest to the phenomenon and who can provide the richest information that can help the researcher formulate theory (Robson 2002). However, in order to have conservative interpretations of the data, the potential for bias in the results was still accounted for in the analysis, for example as shown in § 6.2.2

In addition to the interviews, a total of 49 unique problem solving cases (53 in total) were collected using the structured questionnaire instrument in the last phase of the research, with 71.4% of all cases covering all four avionics systems of interest in this research, while the remaining 28.6% cover several other non-avionics aircraft systems for comparison and verification. Four of the 53 cases were repeat examples given by different subjects and which served to validate as well as add further detail to what was previously reported. The following sections will explore in detail the research methods and practices used to collect and analyze the interview and questionnaire data.

5.2 Research Methods

The majority of research on knowledge integration has been largely conceptual and based on case study investigations, as summarized in Table 1 of Chapter 2 (literature review) under the column labeled "methods". This is a key indication that interviews are a rich method for

collecting data in this area, since research into organizational knowledge and problem solving activities is in essence a study of social phenomena which are best captured through individual and group perspectives. It follows that in order to mine the richness of the insights collected through case interviews in a meaningful way, it is necessary to use systematic and rigorous procedures and methods for organizing and classifying the data into similar groups and conceptualizing relationships between them. One such method is the "grounded theory" approach which is based on the systematic development and refinement of categories and concepts from the collected data in order to build theory, such that the final outcome is a theory that is 'grounded' in the data, as the name of the method itself suggests (Glaser and Strauss 1967; Strauss and Corbin 1990). This means that grounded theory is a largely *inductive* method since conceptualization is done "from the ground up", and since theory is allowed to "emerge" from observation (instead of forcibly fitting higher conceptual notions and ideas down to the data), making it an especially powerful method for studying social phenomena which are not yet well understood, or which are very subjective (i.e. unique to each person or group) or very contextual (i.e. specific to their environment)[28].

In practical terms, the grounded theory approach allows the researcher to go into the field with little or no existing insights, and yet still be able to develop a new and complete theory that frames the phenomenon entirely from observations. This "clean slate" approach has the benefit of foregoing any preconceived biases when the phenomenon is not yet well defined or when it is highly subjective and contextual. However, this does not mean that grounded theory building is not useful for exploring phenomena which have some well defined aspects to them (as in researching a complex phenomenon where some aspects are more or less understood than others). By the same token, this also means that a grounded theory investigation of a complex phenomenon can be guided by, and benefit from, previous insights in the existing literature. Indeed, deduction and verification of basic established principles, while not as emphasized as the inductive part of grounded theory, can nonetheless play an important role in the overall

[28] For example, the grounded theory approach has been especially useful (and thus widely used) in psychological and nursing studies, such as to explain individual behaviors or social processes in dealing with medical issues (Polit and Beck, 2004)

analytical process[29] (Strauss 1987). Such is the case in this research where knowledge integration and problem solving are complex processes which have relatively objective and properly defined aspects to them (in that they are, to a certain extent, already formalized in their organizational context), and therefore drawing on and validating previous theory is necessary to further explore the different aspects of these complex phenomena. Note however that this research started with open unstructured interview questions in order to avoid interjecting previous biases from existing theory into the field observations. In addition, no a-priori codes were used in the subsequent coding of the collected data.

In summary and based on the aforementioned reasons, the grounded theory method was deemed most appropriate for capturing the complexity of the phenomenon at hand and was adopted in this research to first guide the interview process (see § 5.3.1 for an overview of field data collection) and to concurrently perform the analysis on the collected interview data (see § 6.1 for the qualitative data analysis). It is worth noting here that while the term "case studies" is sometimes used in this thesis when describing the separate rounds of field research with different organizations and programs, it is only intended to highlight the fact that multiple grounded theory studies were conducted, so that each study is a separate "case" on its own. This practice is a hallmark of flexible designs overall and is typically done for generalizability purposes (Robson 2002), therefore it is not restricted to the case study approach. This means that for each of the separate cases in this research, the data were collected and analyzed in accordance with the guidelines of the grounded theory method, as summarized in § 5.3.1.

Beyond the field interviews which were used to explore and explain the complexities of the knowledge integration phenomenon itself, a structured questionnaire was also used in the final round of research in order to further frame the dynamics of knowledge integration in its complex setting. The questionnaire allowed the collection of categorical and rank-ordered data which were later used in a separate quantitative analysis to compare the frequencies of outcomes for knowledge integration under different problem solving conditions (i.e. using a comparative

[29] It is noteworthy to point out however that there are two fundamentally opposed views for how to apply the grounded theory method in practice, mirroring differences between the original authors of the method, Glaser and Strauss, where the former advocates the "clean slate" purely-inductive approach whereas the latter adopts a more structured method that is inclusive of deduction and verification techniques. In this thesis, I adopt the latter approach and I draw on insights from the literature to refine the proposed framework for knowledge integration.

analysis of frequency distributions, which is a similar process to hypothesis-testing but without positing hypotheses in advance). The constructs used to collect the questionnaire data are explained in § 5.3.2.

The use of a mixed method approach in this thesis was necessary to capture the dynamic changes in the process of integrating knowledge under non-routine troubleshooting in crisis mode (known as firefighting), as opposed to routine problem solving conditions where the process of knowledge integration is more stable and systematic and therefore less dynamic (refer to Chapters 2 and 6 for a more in-depth discussion of the routine/non-routine dimensions for knowledge integration and problem solving). This additional investigation (beyond the interview part of the research) was considered worthwhile in this thesis since firefighting is a common occurrence in complex product development, and as such the corresponding non-routine mode of knowledge integration is a key aspect of the overall process. It this therefore expected that collecting additional data about the integration of knowledge in non-routine problem solving situations will lead to a more dynamic framework for the overall phenomenon.

In conclusion, the strategy used in this thesis to build theory about knowledge integration is based on combining two types of data collection and analyses, first the grounded theory approach in collecting and analyzing the interview data, and second the comparative analysis approach using frequency distributions and statistical measures to analyze the data collected with the structured questionnaire instrument. Theory development in this thesis is therefore an iterative refinement process of the proposed conceptual framework using both types of data and analyses, as shown in the next chapter on data analysis. But first, an overview of the data collection procedures used in the field research is provided in the following sections.

5.3 Methods for Field Data Collection

As already outlined in Table 26 above, the interview part of the research consisted of multiple site visits to the field in order to collect data and continuously refine the insights derived from the analysis after every round of research. The interview process ended when no new categories of insights were being generated from further research, which is known as *category saturation* in

grounded theory. An overview of the process for collecting the interview data is given in § 5.3.1 below. Once the interview process was completed, a final return visit to the field was done to administer the structured questionnaire using insights from the analysis of the interviews, along with a construct for collecting new data about the frequency of knowledge integration under different problem solving conditions. The process of collecting the questionnaire data is reviewed in § 5.3.2.

5.3.1 Data Collection through Interviews

The strategies and procedures for selecting interview questions and respondents was guided by the general principles of grounded theory research and in accordance with sanctioned practices for qualitative data collection (Strauss and Corbin 1990; Robson 2002). The interview process started out as unstructured and exploratory, with interview questions being continually refined and focused based on earlier observations, such that the questions became much more structured and specific in later rounds of research. This is in line with the *constant comparative* approach to data collection and analysis which is the cornerstone of grounded theory. Similarly, interview subjects were selected before and during the interviewing process with the help of facilitators from the host organizations, where the facilitators helped to identify future interview candidates who best matched emerging needs for further research as identified by the researcher. As a result, questions were increasingly targeted at the right audience as the interview process progressed, which is in accordance with theoretical sampling techniques in grounded theory research. These procedures ensured that the data collection and analyses were done nearly simultaneously, meaning the choice of interview questions and subjects were adjusted and refined in light of emerging findings.

In the first exploratory phase of the interview process, interviews were conducted with personnel from program engineering, program management, functional engineering and material in both the prime and supplier organizations. In addition, where possible, interviews included individuals responsible for knowledge management or familiar with the major initiatives for knowledge transfer and sharing in their respective organizations. The interviews at this stage amounted for the most part to an open dialogue about the "big picture" of knowledge integration

within each organization and with external partners and suppliers, with some directed questions intended to focus the discussion around the most frequently used channels, strategies, practices and mechanisms for exchanging knowledge, as well as the most commonly encountered barriers impeding knowledge flow and the kinds of enablers that the organization had in place to overcome those barriers. As the interview process progressed, the exploratory interview questions were revised to focus on the most important aspects of the knowledge integration phenomenon, and to collect more data about the mechanics of knowledge integration. This was done by revisiting the same individuals to explore previous insights in more detail, or by selecting new subjects who can provide new insights from different organizational perspectives. Refer to Appendix A.1 for sample exploratory questions and Appendix A.2 for sample focused questions used in the different phases of the interview process.

5.3.2 Data Collection through the Questionnaire Instrument

In order to fully capture the dynamics of knowledge integration in a complex setting such as that of complex avionics development being researched in this thesis, it is necessary to account for complexity factors that most affect the choice of strategies/practices and channels/mechanisms for integrating knowledge under non-routine conditions. Based on established insights and deductions from the literature as presented in Chapter 2 of this thesis, the factors of most influence on the process of integrating knowledge in large-scale complex product development environments are the type of architecture for the product under development, the degree of newness of the underlying technology (or core technology) embedded in the product, and the type and novelty of the problem for which knowledge is being integrated in support of the problem solving effort. An aggregation of those main factors can then be considered as an estimate measure of the overall complexity of the problem solving situation. Therefore, it is necessary to collect measurable data about these contextual factors in order to determine their individual and collective effects on the process of integrating knowledge in complex product development.

Note that there are other complexity factors of influence over the knowledge integration process under non-routine conditions beyond the product-centric and problem-specific factors mentioned

above, notably those aspects related to the organizational environment (such as the number of stakeholders involved in the problem solving effort and the types of boundaries crossed during problem solving) which also affect the choice of strategies and mechanisms for integrating knowledge. However, organizational factors are implicitly accounted for in the proposed framework for knowledge integration (i.e. the stakeholders involved in problem solving are also involved in the knowledge integration effort) therefore it is not necessary to collect organizational-specific data in the questionnaire. As such, the appropriate complexity measures needed for framing the knowledge integration process under non-routine conditions can be limited to the product- and problem-specific measures mentioned above.

From a data collection perspective, while it is possible to ask respondents directly for their evaluation of the actual type and novelty of the problem or to even deduce these measures indirectly from their description of the problem, it is much more tenuous to rely on their recollection and evaluation of product-specific complexity factors. This is because respondents to the questionnaire are asked to describe problems they are closely involved in, and therefore it can be assumed they have intimate knowledge about problem-specific details and that their recollection of those details will be sufficiently reliable. However, this may not be the case for system-specific details since their knowledge of such details might not be as intimate (e.g. the knowledge of a system integrator about the internal workings of the subsystem affected by the problem is much more limited than the supplier's knowledge of that subsystem). It is also more reasonable to expect that the recollection of system-specific details might be more limited or even restricted due to common proprietary or classified technology reasons. For example, it is ideally most desirable to collect accurate information about the internal characteristics of a product in order to measure its complexity level, such as collecting data about the architecture of the system where complexity is measured as the "total number of interconnections between different parts of the system" (Moses 2002), or as the ratio of the "total number of components" over the "total number of functions" delivered by the system (Ulrich 1995). By these measures, the level of system complexity is a function of the degree of interdependence or coupling between its constituent parts[30]. Alternatively, and since all of aforementioned measures require

[30] The number of interconnections (or interfaces) in a product is an indication of the degree of modularization (or conversely the degree of integration) in the product's architecture, with higher interconnections indicating a more

detailed knowledge about the layout of the system which was not be available or possible to collect in this research, a third type of complexity measure that was considered is a combination of "the number of parts in the system" and "the degree of modification of the system design relative to a previous design", where a new design is more complicated than a modification and where a major modification is more complicated than a minor modification (Takeishi 2002). Taken separately, each measure is an evaluation of system complicatedness as opposed to system complexity (as already explained in § 2.3.1), but combined together, they can indicate the relative complexity of the system where the problem is rooted.

It is arguably easier to collect the appropriate data for these last two measures, however, due to the highly classified nature of military avionics and the deep proprietary knowledge involved, it was not possible to collect any of the previous types of data in the field or to obtain publicly available information from the literature with enough specificity and detail to allow the use of the measures discussed above. Instead, a surrogate measure of system complexity was constructed for the purposes of the questionnaire, based on problem-specific information that the questionnaire was already designed to collect such as "the total number of systems affected by the problem", including avionics and non-avionics systems (measured on a scale of 1 to 5), combined with the degree of maturity (or newness) of the core technology of the affected system or systems (measured with the categorical variables "High" and "Low") and the degree of problem novelty (measured with the categorical variables "New" and "Old"), where an old problem is one which has been previously solved or to which the solution steps are largely known (e.g. where the root cause of the problem has already been identified, or where the solution steps are known for the most part)

Aggregated together, these measures indicate the degree of problem complexity which can be used to differentiate between different troubleshooting conditions and therefore to evaluate the corresponding differences in the use of knowledge integration channels and mechanisms. The above measures were collected either directly from the respondents or indirectly by deducing the information from the descriptions of the problem and the problem solving approach. To ensure

integrated architecture. Similarly, the number of system components delivering the same function is an indication of the degree of coupling between these components, with a high number of components tied to the same function indicating a tightly coupled layout.

reliability of the responses, the population for the questionnaire respondents was limited to IPT leads and engineers who were directly involved in solving major problems in the development of the four mission avionics systems of interest. To further ensure comparability among the collected data, only technical problems related to subsystem engineering and system integration issues were considered. This is in line with the initial focusing of the research on the "System Design Phase" of the development process where these types of problems are most common.

The questionnaire used to collect the above data is shown in Appendix A.3. Figure 22 below illustrates the full sample of 49 problem cases, shown by the program and the corresponding organizations where the data was collected from.

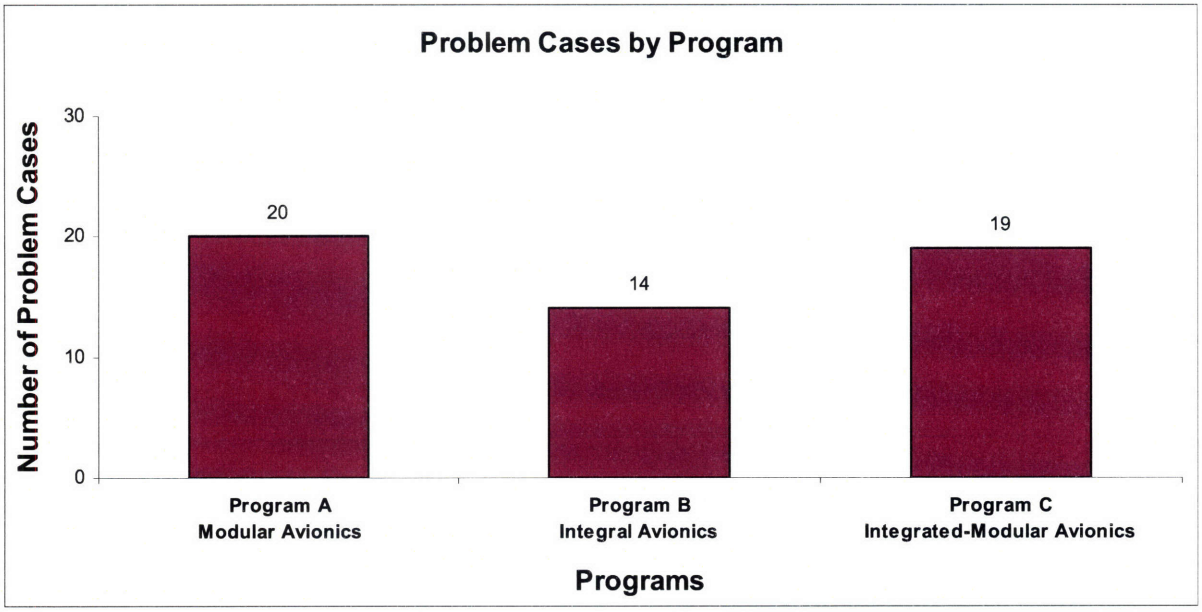


Figure 22: Overview of Problem Cases Collected by Program and Organization

Note that there are more problem cases provided by the prime organization than by the suppliers. However this is not considered to be a source of bias in the analysis since avionics IPT's, which are the main locus of problem solving in all problem cases, include representatives from the supplier organizations, thereby ensuring that problem solving perspectives are not skewed in favor of the prime organization.

6. DATA ANALYSIS

This chapter is divided into two parts: the first part is a qualitative analysis of the exploratory and semi-structured interview data collected across all three case studies and which serves to identify the main characteristics of the knowledge integration process in a large-scale complex product development context, with the purpose of refining the original framework for knowledge integration proposed in Chapter 3 of this thesis; the second part is a quantitative analysis of the survey data obtained through the structured questionnaire administered across all three case studies and which serves to further frame the characteristics of knowledge integration in different complex problem solving contexts.

6.1 Qualitative Analysis of the Interview Data

The qualitative case study analysis in this section is based on the grounded theory method introduced in Chapter 5 of this thesis. The data used in this analysis include the exploratory and semi-structured interview data collected across all three case studies, but does not include the structured questionnaire data collected in the final round of interviews and used for the quantitative part of the analysis presented in § 6.1.2 below[31]. The qualitative analysis was done using a series of procedures for data reduction, visualization and interpretation (Miles and Huberman 1994; Robson 2002), as follows: 1) an initial manual categorization of the interviews in order to organize and classify the 'raw data' along major categories and subcategories of channels, mechanisms, strategies and practices for knowledge integration in a large-scale complex product development context; 2) a computer-assisted coding of the thematic data along dimensional properties for each identified category using the specialized software tool "**MAX**QDA 2007"[32] and in accordance with guidelines for grounded theory coding (Strauss and

[31] Specifically, the data used in the qualitative analysis include the interview responses from the first three site visits to the prime organization (Company A) and the first two rounds of interviews conducted with the supplier organizations (i.e. the first two site visits to Company B and the first two series of teleconference calls with Company C). In addition, general insights have been extracted from the last round of interviews where the structured questionnaire was administered at all three organizations; however the questionnaire data was not used in the qualitative analysis since these data are specific to particular problem solving contexts.

[32] The software tool "**MAX**QDA 2007" (which stands for Maximum Qualitative Data Analysis Version 2007 Release 060607) is described as "a state-of-the-art instrument for professional text analysis…one of the pioneers in the field (the first version was released in 1988)…a global leader for computer assisted qualitative data analysis" (reference: **http://www.maxqda.com** accessed September 14, 2007)

Corbin 1990); and, 3) interpretive and matrix analyses of the coded and categorized data to visualize and identify patterns and relationships within and across categories and subcategories (see (Robson 2002) Chapter 14 for a detailed discussion of the interpretive/grounded theory and matrix approaches for qualitative data analysis). Insights from existing theory about knowledge integration as reviewed in Chapter 2 of this thesis were used to guide the coding process and the overall analysis of the data. The intermediate and final results of the qualitative analysis are presented and discussed separately in the following subsections.

6.1.1 Main Characteristics of Knowledge Integration

As already outlined in § 5.3.1 on the methods used for collecting data through interviews, the qualitative data in this research were collected over several rounds of field interviews and a preliminary analysis was done after each round in order to a) classify and interpret the newly collected data, b) refine previous interpretations and c) adjust interview questions for subsequent rounds in light of emerging findings. Once all the data collection was completed, a full qualitative analysis was performed again from scratch on the complete set of data from all three case studies. The steps of this analysis are described in this section.

First, an initial reduction of the interview data was done manually by going through the original interview transcripts from each case study and identifying and labeling "critical instances" of channels, mechanisms and strategies/practices for knowledge integration, where a critical instance is the result of repeated or emphasized information from multiple interviews within and across case studies[33]. This was followed by a second step of classifying related instances together in groups of channels, mechanisms and strategies/practices to form the main characteristics of knowledge integration. Per the guidelines of the grounded theory method, data reduction and classification was conducted nearly simultaneously with the collection of the data in each case study, and then again at the end on the complete set of data collected from all case studies. This ensures the relevance of the final classifications and adds to the validity of outcomes and interpretations later in the analysis (Robson 2002). In total, this process allowed

[33] Recall that the primary motivation of this research as outlined in § 1.2 of this thesis is to frame the 'mechanics' of knowledge integration in terms of its channels, mechanisms and strategies/practices (as defined in § 2.1.3).

the identification of 8 unique channels, 67 strategies and practices (40 of which are unique while the remaining 27 are repeated strategies/practices over multiple channels) and 114 mechanisms (79 of which are unique). In addition, there were also 56 types of barrier situations across all eight channels (44 of which are unique) and 29 enabling conditions (22 of which are unique) that were also identified in the initial data reduction and classification phase[34]. The barriers and enablers can be considered as "blockers" and "un-lockers" of knowledge integration channels and serve to provide context for the analysis. For reference, all of the above mentioned knowledge integration characteristics are presented in Tables 41 to 48 of Appendix B.

In order to make inferences from the classified data, a matrix analysis was performed to visualize and identify patterns and relationships within and across groups. The data were first summarized to find the most dominant characteristics of knowledge integration, which were then divided by knowledge integration modes (specifically the routine and non-routine modes) and phases of integration (namely the knowledge transfer, sharing and application stages). This approach was guided by the main insights obtained from the literature as summarized in § 2.4, which predicated a differentiation in the process of integrating knowledge under routine versus non-routine conditions (i.e. during routine product development work versus non-routine troubleshooting events)[35]. This is based on the fact that large-scale organizations integrate knowledge using both routine and non-routine strategies, practices, channels and mechanisms in order to support both routine and non-routine problem solving. This approach to the analysis is in line with the central argument advanced in this research that problem solving is the thread through which the framing of the knowledge integration process can best be accomplished, and as such any proposed framework for knowledge integration must account for different problem solving contexts. Using the above dimensions, the dominant characteristics for knowledge integration are presented below in Tables 27 and 28 for routine and non-routine cases, respectively.

[34] Note that the labeling and classification of the data as discussed here is different from "conceptual labeling" and "conceptual categorizing" in grounded theory development, which are discussed separately in § 6.1.2. This is because the labels and groups developed in the initial data reduction phase are mostly straight summaries and consolidations of the raw data, and as such there was no further conceptualization of the data at this stage.
[35] As posited in the literature, in a crisis there is a switch from routines to group problem-solving, with consensus decision-making and brainstorming being the most useful for unusual, complex tasks (Grant 1996a).

Table 27: Routine Knowledge Integration in Large-scale Complex Product Development

Integration phase	Channel	Strategies / Practices / Mechanisms
Knowledge transfer	Intra-program	Individual and program training, concurrent work tools, program templates, libraries, databases
	Program-program	Common suppliers, shared databases, design transfer, process transfer, technology transfer
	Function-program	Boundary spanners, specialized training, job rotation, process standardization
	Prime-supplier	Shared databases, design transfer, process transfer, technology transfer
Knowledge sharing	Intra-program	Lessons learned share sessions, conferences, personal and group communication
	Program-program	Communities of practice, multi-program share sessions, personal communication
	Function-program	Process improvement, personal and group communication
	Prime-supplier	Conferences, process improvement, tight partnerships, supplier management, personal and group communication
Knowledge application	Intra-program	Design and program reviews (gold team reviews), intra-IPT problem solving and status update meetings
	Program-program	Non-advocate design reviews, participating in gold teams
	Function-program	Independent design reviews, participating in gold teams
	Prime-supplier	Design reviews, early supplier integration, co-location

As the above table illustrates, routine knowledge integration involves a higher number and variety of strategies, practices and mechanisms for transferring and sharing knowledge, while those for applying knowledge are more limited in scale and scope. This is because the integration of knowledge in this case serves to support routine work where efficient access to existing knowledge resources is what is usually required to carry out normal tasks. To further analyze the main characteristics of routine knowledge integration, a down-selection to the most commonly cited strategies/practices and mechanisms for integrating knowledge in routine problem solving contexts was performed, and the resultant categories are depicted visually in the following figure 23 below:

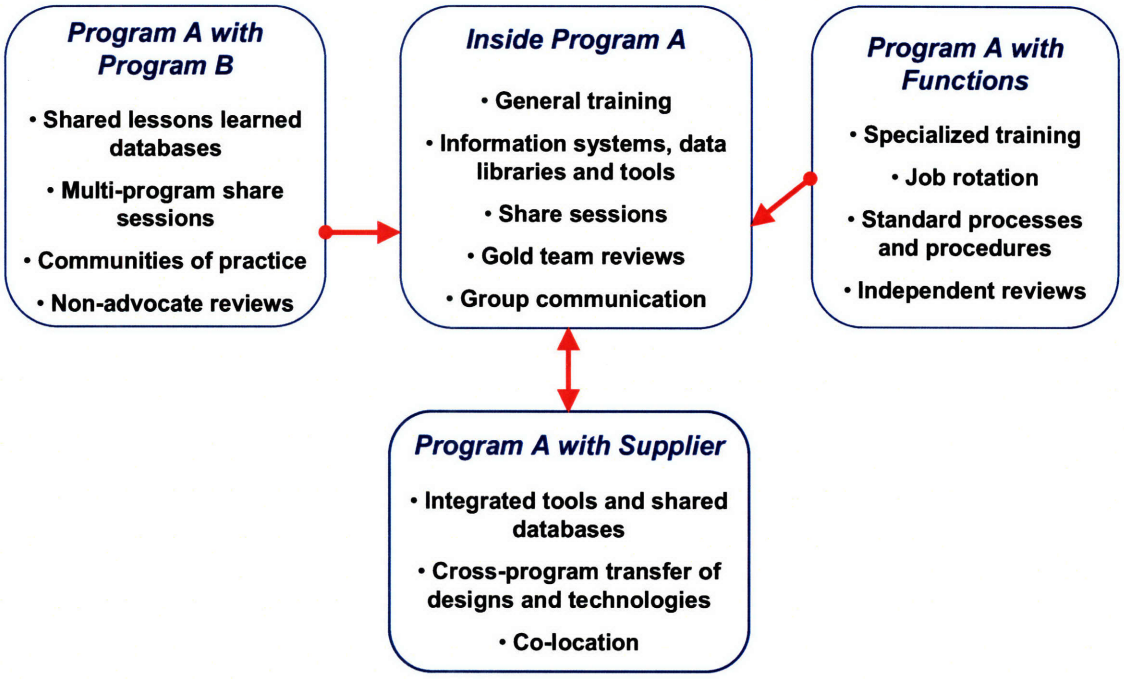


Figure 23: Routine Knowledge Integration in Large-scale Complex Product Development (Relationships Shown Relative to Program A)

A key takeaway from the previous figure is in the apparent emphasis on the sharing of lessons learned from past problem solving events across multiple channels and through a variety of mechanisms (e.g. share sessions, shared lessons learned databases). This is due to the fact that this strategy allows the transfer of rich knowledge content that is necessary for dealing with complexity, which makes it a notable aspect of knowledge integration in complex product development.

Key takeaway:
Sharing lessons learned from previous problem solving events is an important strategy for dealing with complexity in product development

It is also clear from the interview data tabulated above that there are other dominant strategies and mechanisms for routine integration which are common across multiple channels, and which

can be summarized as: 1) training, 2) databases, 3) design reviews and 4) the transfer of people and information across various boundaries. These are in addition to personal and group communication mechanisms which are the cornerstone of knowledge integration along all organizational channels, with emails and staff meetings constituting the majority of those interactions in the routine context. It can, therefore, be concluded that a significant number of mechanisms and practices in the routine case are designed to integrate explicit knowledge in the most efficient way. However, it should be noted that there were several drawbacks associated with this efficiency approach that were frequently cited by interviewees, and the main takeaway from these relates to the use of information repositories such as databases and data libraries, which serve the efficiency purpose in the integration process at the expense of effectiveness in the final outcome. Specifically, databases were unanimously reported as being difficult to search (due to information overflow) and rarely trusted (due to information obsolescence and lack of validation). As a result, these resources are rarely used,[36] as widely reported by the vast majority of interviewees, and there is a common perception among engineers that the information stored in databases is mostly "written by technical people who have nothing better to do and just want to make a point". This explains why cross-program knowledge integration (i.e. between program A and program B) was frequently described as the proverbial "weakest link" by a majority of interviewees, as this channel is dominated by ineffective databases, which is also compounded by the low attendance of multi-program share sessions, especially by senior engineers who are focused on their own internal program goals. As one interviewee put it bluntly, "the biggest barrier to knowledge integration (in complex development) is in the program-to-program interface", since programs tend to be internally focused and are frequently described as "silos" and "fishbowls".

Key takeaway:

The program-to-program channel is the "weakest link" for integrating knowledge due to the dominance of explicit knowledge integration mechanisms which are ineffective

[36] In a knowledge management survey conducted by Supplier C of their product development and engineering personnel, a majority of respondents reported "rarely" using data libraries, and almost 70% of respondents reported great difficulty in locating appropriate lessons learned through information systems.

However, this does not mean that all interactions involving codified knowledge are perceived in the same way, in fact most interviewees agreed that accessing and using previous solutions, for example those captured in the form of standard methods, checklists and templates in program data libraries is a frequently used strategy for integrating knowledge during routine development work, which makes this type of knowledge greatly valued since it consistently affords significant savings in time and effort that would otherwise have been spent rediscovering the same solutions. In general, it can be concluded from the foregoing analysis that the dominant types of mechanisms and strategies employed in the routine integration of knowledge consist for the most part of periodic activities and long term strategies geared towards efficiency of integration, which is reflective of the relatively stable problem solving environment. This is in contrast to the non-routine problem solving context where knowledge integration requires rapid mobilization capabilities, as shown in Table 28 below.

Table 28: Non-Routine Knowledge Integration in Large Complex Product Development

Integration process	Channel	Strategies / Practices / Mechanisms
Knowledge transfer	Intra-program	--
	Program-program	Moving subject matter experts, tech fellows and gurus
	Function-program	Deploying boundary spanners, moving subject matter experts and tech fellows
	Prime-supplier	Moving subject matter experts, site visits
Knowledge sharing	Intra-program	Asking gurus
	Program-program	Asking counterparts
	Function-program	Asking graybeards
	Prime-supplier	--
Knowledge application	Intra-program	Taskforces, tiger teams, red team reviews, working with wizards, subject matter experts and tech fellows, inter-IPT problem solving meetings
	Program-program	Participation in taskforces, tiger teams, red teams
	Function-program	Participation in taskforces, tiger teams, red teams
	Prime-supplier	Participation in taskforces, tiger teams, red teams

Compared with the routine problem solving context, there is a greater emphasis in the non-routine case on the use of special strategies, practices and mechanisms for applying knowledge rather than for transferring and sharing it. This is because knowledge integration in this case serves to support unusual problem solving situations requiring live troubleshooting and group collaboration rather than mere access to knowledge. Figure 24 below illustrates the main categories tabulated above.

Figure 24: Non-Routine Knowledge Integration in Large-scale Complex Product Development (Relationships Shown Relative to Program A)

A close examination of the interview data depicted here shows that non-routine knowledge integration is dominated by 1) deployment of experts across boundaries, 2) consultations with experts in other programs and functions, and 3) formation of special teams (meaning temporary teams grouping experts from different organizational levels and affiliations and tasked with diagnosing and/or solving a specific problem). It is thus clear that the dominant types of knowledge integration mechanisms and strategies employed under non-routine conditions consist for the most part of short term activities and special interventions, indicating the need for quick response and flexible organizational capabilities. It is also evident from the previous table that

tacit knowledge interactions (within special teams and with outside experts) dominate the integration process in the non-routine case.

> Key takeaway:
>
> Major non-routine problems in complex development require extensive tacit knowledge integration through mobilization of experts and forming special multi-disciplinary teams

Another notable insight from the previous figure is that functional groups play a major role in deploying experts and forming special problem solving teams in non-routine problem solving situations. This is in contrast to the role of functional groups in routine work where functions are mostly involved in the integration of codified knowledge to-and-from programs in the form of standardized methods and processes, while the tacit knowledge integrated along this link in routine situations (e.g. job rotation and training of individuals) was, for the most part, not directly related to specific problem solving needs. The main channel for integrating tacit knowledge in the routine case is the prime-supplier link where specialized knowledge and new information and technologies are transferred, shared and applied in routine problem solving situations.

Finally, it is important to note that the main purpose of the previous data tabulation is to illustrate the findings from the field research, and is not intended to provide an exhaustive or complete listing of all possible strategies or mechanisms that may be useful for integrating knowledge in a large-scale complex product development context, but rather only those strategies and mechanisms that were discovered and emphasized in the course of the field research. I also note that while the focus of this thesis is on knowledge integration for technical problem solving, the data summarized in these tables include findings about the integration of non-engineering knowledge (such as non-technical process knowledge), as well as about the integration of knowledge outside the context of problem solving (such as process improvement). These findings, which are outside the scope of the thesis, are included for completeness and reference only but are left out of subsequent steps in the qualitative analysis.

Integrating the findings above, the knowledge integration process can be described at a high-level of abstraction for all problem solving cases (both routine and non-routine) as follows:

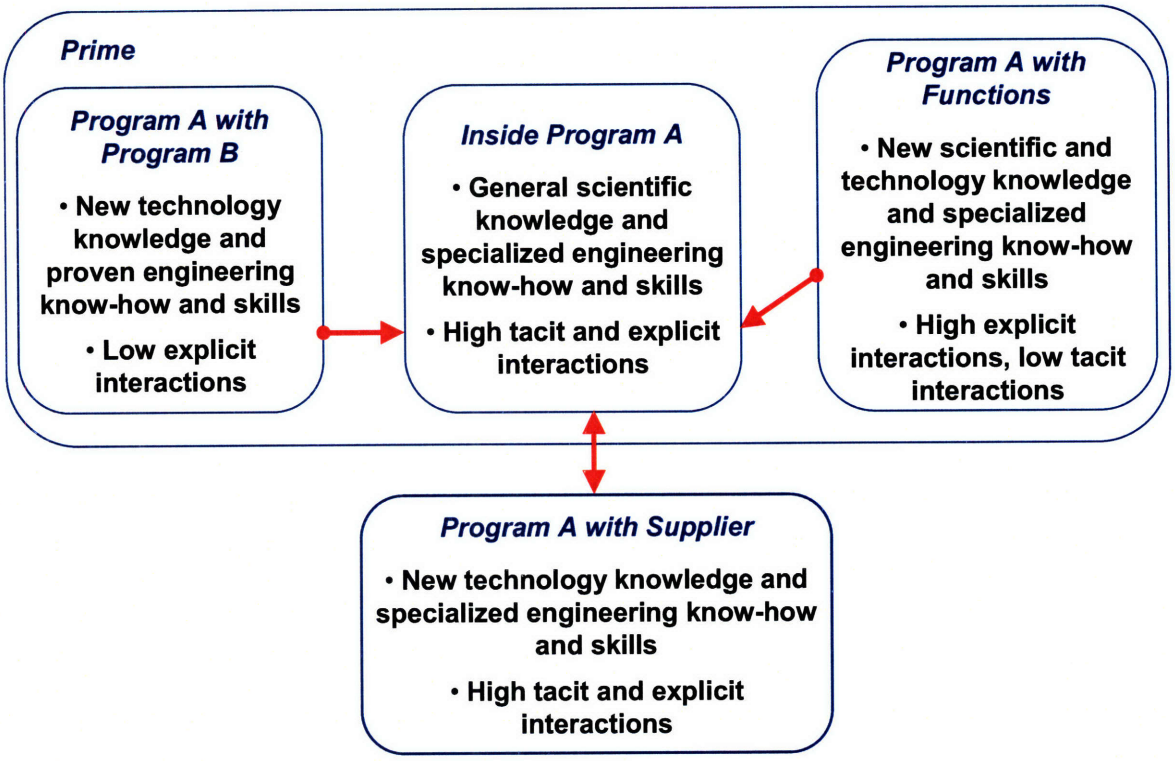


Figure 25: Preliminary Empirical Framework for Knowledge Integration in Large-scale Complex Development (Relationships Shown Relative to Program A)

The figure above shows that functional groups are the source of new knowledge in terms of expertise and new information related to scientific disciplines and technologies (e.g. new scientific knowledge about radio wave frequency compression) as well as specialized know-how and skills about the design and development of complex systems (e.g. new engineering knowledge about radar array design). On the other hand, cross-programs interactions are useful for integrating proven know-how and skills in subsystem design and integration, while suppliers are the main source of technological innovation and deep expertise in the latest design and development concepts for subsystem modules. Therefore, and per the preliminary findings from this stage of the research as illustrated in the above framework, I conclude that new technology knowledge and design expertise are integrated mostly across vertical and horizontal channels,

while new scientific knowledge and proven analytical skills are integrated along lateral channels internally to the single firm. In addition, the framework shows that technological innovation can be integrated across multiple channels, however as already shown in Figure 23 above the primary source for new technologies are the suppliers common across programs and who act as a 'funnel' for indirect knowledge flow between programs.

6.1.2 Concept Development and Theory Building

The matrix analysis in the previous section served to refine the proposed conceptual framework in Chapter 3 of this thesis. However to further build and expand on existing theory requires a *conceptualization* of the data in order to discover new concepts and relationships related to the phenomenon of knowledge integration. Since this part of the analysis relies on purely qualitative interview data, the rigorous procedures of the grounded theory method were used to develop concepts out of the classified data (Glaser and Strauss 1967; Strauss and Corbin 1990; Robson 2002). This procedure is known as "open coding" in the methods literature and consists of identifying and naming actual phenomena using higher-order conceptual labels, and then *categorizing* similar concepts together (Strauss and Corbin 1990; Pandit 1996). This requires a constant comparison between different phenomena and a continual refinement of the conceptual labels as well as constant updating of the identified categories. Note that this step is described separately from the previous data reduction steps for clarity only; however, it was not conducted entirely separately or sequentially after the first steps of organizing and classifying the data, but instead concepts were developed in conjunction with the reduction of the data after every round of interviews, and continually refined/updated where appropriate.

The results of the conceptualization procedure are presented in Tables 29 through 33 below. The developed concepts are listed in the first column of each table, with each concept being derived from abstracting a number of strategies/practices and mechanisms listed alongside it on the same row. The rows thus reflect the connection between a particular strategy for integrating knowledge (listed in the second column) and the corresponding mechanism(s) for implementing it (shown in the third column).

Table 29: Intra-Program Knowledge Integration (Along Channels #1 and #2 in Conceptual Framework)

Conceptual Labels	Strategies / Practices	Mechanisms
Advice sharing by experts	Calling on program experts	Asking the "gurus" (gurus are very knowledgeable senior staff with long experience but who are not directly involved in problem solving)
Advice sharing by social networking	Informal apprenticeship	Buddy system (social pairing for sharing general – not technical – experiences between senior and junior staff)
Advice sharing by special review teams	Routine team reviews	Gold teams (special teams of experts for doing routine design and zone reviews to identify problems in design and at interfaces, such as at preliminary and critical design reviews)
	Non-routine team reviews	Red teams (special teams of experts to review major problems and provide non-binding suggestions for problem solving)
Assistance by experts	Joint problem solving with program experts	Working with the "wizards" (wizards are experts with long experience in a disciplinary subject, for example radar signatures, they work with product teams to fix problems in the lab)
		Working with the "subject matter experts" (SME's have deep expertise with a particular system, such as the radar system; they are deployed on programs and involved in direct problem solving with product engineers)
		Working with the "tech fellows" (tech fellows have deep knowledge in 1 of the 3 major areas, namely hardware, software and systems engineering, they work on programs with product teams and serve only to troubleshoot tough problems)
Assistance by group collaboration	Joint problem solving by routine team meetings	Brainstorming meetings (investigate/solve problems with other teams)
		Root-cause analysis meetings (informal communication among different functions/disciplines to troubleshoot major problems)
Assistance by special action teams	Joint problem solving by non-routine team meetings	Tiger teams (special teams of experts to diagnose/solve major problems)
		Taskforces (large multi-disciplinary and broad membership teams to tackle critical system-level issues)
Information transfer by boundary objects	Integrated concurrent engineering	Prototypes, mockups, models, simulations, demos to visualize the entire design and locate potential problems

Table 29 – Continued: Intra-Program Knowledge Integration

Conceptual Labels	Strategies / Practices	Mechanisms
Information transfer by boundary objects	Integrated concurrent engineering	Integrated work tools (for concurrent modeling, drafting, design, workflow management)
Information transfer and sharing by IT infrastructure	Routine knowledge capture and reuse	Program data libraries (electronic repositories for programmatic information, software code)
		Program templates/checklists
		Lessons learned databases (electronic repositories of past knowledge)
	Routine formal communication	Work documents flow (e.g. memos, manuals, reports, logs, test data, requirements/specification documents)
		Emails, netmeetings, phone calls, teleconferences, videoconferences
Information sharing by group communication	Routine team meetings	Status meetings (formal/semi-formal presentations to update other staff with new information)
	Regular conferences	Monthly knowledge symposium
	Formal sharing of lessons learned	"Share sessions" (face-to-face or teleconference meetings for exchanging information based on past experience, for example stories of previous problem solving events)
	Informal sharing of lessons learned	Brown bag sessions (ad-hoc share sessions over lunch to hear about other people's problems)
Information sharing by personal communication	Routine informal communication	Walk-and-talk (oral communication of information, experience and lessons learned with peers)
Information sharing by training	Formal program training	Familiarization process (familiarizes new hires with program operations, acronyms)
Information, know-how and skills sharing by training	Formal individual training	Mentoring (periodic sharing of technical expertise between senior and junior engineers)
Information, know-how and skills sharing by group communication	Formal team structuring	Integrated product team structure (IPT structure co-locates multiple disciplines, integrates different individual and organizational expertise)

Table 30: Program-to-Program Knowledge Integration (along Channel #3 in Conceptual Framework)

Conceptual Labels	Strategies / Practices	Mechanisms
Advice sharing by personal communication	Informal non-routine individual communication	Talking to colleagues (call-up of counterparts in other programs)
Advice sharing by special review teams	Routine team reviews	Non-advocate design reviews by one program for another (staff from one program participate in gold team reviews of another program)
	Non-routine team reviews	Mixed program participation in special review teams (staff from other programs assigned to gold teams, red teams)
Assistance by moving experts	Joint problem solving with program experts	Loan out subject matter experts, tech fellows and gurus to another program (short and long-term assignments)
Assistance by special action teams	Joint problem solving by non-routine team meetings	Mixed program participation in special action teams (staff from other programs assigned to tiger teams, taskforces)
Information transfer by commonality	Indirect design, technology and process transfer	Using common suppliers with other programs
		Prototyping new system designs, architectures and technologies by older programs for future ones
Information transfer and sharing by IT infrastructure	Shared information systems	Shared repositories (shared data libraries, shared intranets)
		Common repositories (multi-program lessons learned database)
Information sharing by experts	Formal sharing of lessons learned	Gurus from one program share lessons with another program
Information sharing by group communication	Routine team meetings	Mixed-program meetings (personnel from one program routinely attend status meetings in another program)
	Formal sharing of lessons learned	Multi-program share sessions
		Chief engineers periodic meetings to share problems, lessons
		Program managers/chief engineers/team leads from older programs convene to share lessons at start of new program
		White papers to notify other programs of problems with potential effect on them (e.g. problem in common subsystem)

Table 30 – Continued: Program-to-Program Knowledge Integration

Conceptual Labels	Strategies / Practices	Mechanisms
Information transfer by boundary objects	Direct design, technology and process transfer	Transferring new technologies, engineering and manufacturing discoveries and materials between programs
		Transferring new design and architectural concepts between programs
		Transferring processes and procedures
Information, know-how and skills sharing by professional networking	Professional affinity groups	Communities of practice (integrate common expertise across programs)
		Working groups (integrate common specialties and subspecialties, such as the avionics working group)
		Focus groups (grouping by common interest, e.g. systems engineering focus group)

Table 31: Function-to-Program Knowledge Integration (along Channel #4 in Conceptual Framework)

Conceptual Labels	Strategies / Practices	Mechanisms
Advice sharing by experts	Calling on function experts	Asking the "graybeards" (graybeards are not deployed to programs, they are retired executives and senior staff with long experience, they do big-picture reviews and evaluations of best solution path)
Advice sharing by special review teams	Routine team reviews	Independent design reviews / walkthroughs (several gold team reviews per year to help solve problems and capture new knowledge)
		Information audits (e.g. audits of technical standards by graybeards, audits of processes and procedures)
		Program reviews (e.g. periodic reviews of system design issues, capability reviews to capture "concept-of-operations", team accountability reviews to elevate problems to higher level)
	Non-routine team reviews	Functions form and participate in special review teams (red teams) to review major problems (functional groups are responsible for picking/negotiating the membership of these teams)
Assistance and information sharing by boundary spanners	Supporting cross-boundary communication and joint problem solving	Deploying "liaison engineers" (liaison engineers serve as conduits between different groups)
		Deploying "material integrators" (integrators serve as conduits between functions and programs)
Assistance by moving experts	Joint problem solving with function experts	Functional groups loan out "tech fellows" to programs for specific problem solving events
Assistance by deploying personnel	Joint problem solving with function personnel	People deployment, long-term assignments (functions deploy people to programs, decide who works on what program or team by forecasting program skill needs and hiring/matching right skills to needs)
		People deployment, short-term assignments (e.g. function engineers help programs directly with problem solving, e.g. helping to assess stability issues, helping with root cause analysis)
Assistance by special action teams	Joint problem solving by non-routine team meetings	Functions form and participate in special action teams (tiger teams, taskforces) to tackle major problems (functional groups are responsible for picking/negotiating the membership of these teams)

Table 31 – Continued: Function-to-Program Knowledge Integration

Conceptual Labels	Strategies / Practices	Mechanisms
Information sharing by boundary objects	Supporting integrated concurrent engineering	Functional groups deploy integrated work tools (e.g. integrated management tool, common product design and management systems)
Information sharing by commonality	Supporting commonality across programs	Commonality initiatives (e.g. initiatives for common designs and architectures across programs)
Information sharing by IT infrastructure	Supporting cross-program communication	Functional groups deploy shared-access information systems (e.g. intranets, shared databases, expertise locator systems)
		Functional groups share generic information across programs (e.g. material group shares all supplier information with all programs)
Information sharing by standardization	Supporting standardization across programs	Functional groups standardize processes and procedures across programs (e.g. standard software development process, design templates)
	Sharing lessons learned	Functional groups capture and disseminate lessons learned across programs (e.g. checklists for typical problems)
	Routine knowledge capture and dissemination	Functional groups develop and disseminate technical manuals and documentation across programs (e.g. software development manual, systems engineering guide)
Information sharing by training	Formal function training	Technical courses (e.g. system design courses taught across programs)
		On-boarding (e.g. training and familiarization for new hires with tools, processes, acronyms)
Information, know-how and skills sharing by training	Career development	Job rotation (personnel moved around functions and programs to gain broad skills, e.g. through the leadership development program)
Information transfer by organizational coordination	Supporting process improvement	Continuous improvement (dedicated office for transfer of Lean principles to programs)
		Initiatives to reduce information overflow by summarizing knowledge
		Functional groups capture and share best practices
		Functional groups integrate dispersed competencies and skills (from handshake process to close coordination)

Table 32: Prime-to-Supplier[37] Knowledge Integration (along Channel #5 in Conceptual Framework)

Conceptual Labels	Strategies / Practices	Mechanisms
Advice sharing by group communication	Routine team meetings	Joint product assessment teams (knowledge sharing node with suppliers, a forum grouping prime and supplier experts to share ideas)
Assistance by moving personnel	Joint problem solving with supplier personnel	Supplier co-location (personnel deployment for long term collaboration on-site, e.g. at RFP stage to help define requirements based on supplier capabilities, or at design stage to give/get help with problem solving)
Assistance by moving experts	Joint problem solving with supplier experts	Site visits (short term collaboration on-site to troubleshoot problems)
		Assignment (short-term) of experts (prime sends out engineers/experts to supplier to improve design process, help prevent future problems)
Assistance by special action teams	Joint problem solving by non-routine team meetings and reviews	Mixed prime-supplier participation in special action teams (red teams, tiger teams, taskforces)
Assistance by group collaboration	Joint problem solving by routine team meetings and reviews	Decision boards (technical decision board grouping prime and supplier experts, decide and approve problem solving approach)
		Working groups (strategic supplier group for working out supplier problems across programs, coming up with common solutions)
Information sharing by boundary objects	Shared information systems	Shared repositories (requirements management database, common problem reporting system, risk database, product data management system)
	Integrated concurrent engineering	Concurrent design and management tools (e.g. integrated computer aided engineering tools, common risk management tool)
Information transfer by boundary objects	Direct design, technology and process transfer	Outsourcing and in-sourcing new technological knowledge (e.g. transfer of new materials discoveries from prime to supplier; supplier reveals proprietary design knowledge to prime during problem solving)
Information sharing by commonality	Single sourcing	Using the same supplier across programs (fosters open sharing across programs, increases supplier cooperation and information coordination)

[37] The label "prime-to-supplier" is not meant to indicate a unidirectional flow of knowledge from prime to supplier; in fact knowledge integration between prime and supplier is bi-directional.

Table 32 – Continued: Prime-to-Supplier Knowledge Integration

Conceptual Labels	Strategies / Practices	Mechanisms
Information sharing by commonality	Indirect design, technology and process transfer	Design or module commonality across programs (transferring designs or buyer furnished equipment from supplier to multiple programs, requirements commonality across programs)
Information sharing by IT infrastructure	Routine formal communication	Emails, phone calls, net-meetings, memos, contracts, technical agreements, teleconference and videoconference calls for status updates
		Work documents flow (requirements docs, interface control docs, drawings, failure reports, engineering change requests/orders, design evaluation/change requests, system architecture charts, test docs)
Information sharing by group communication	Routine team meetings	Technical meetings at prime's location to share information about system level problems
		Technical interchange meetings at supplier location to share information about lower-level issues
	Regular conferences	Annual supplier conferences, monthly symposiums (for sharing information about problems, progress)
	Formal sharing of lessons learned	Special communication forums (strategic supplier advisory group for sharing lessons learned)
Information sharing by personal communication	Informal routine individual communication	Ad-hoc sharing of lessons learned in meetings, discussions and reviews
Information transfer by network cooperation	Process improvement	Transfer of best practices to/from supplier
Information transfer by network coordination	Supplier management	Carrot-and-stick management (leveraging large size of contracts and future program opportunities to push for open sharing by suppliers)
Information, know-how and skills sharing by network collaboration	Early supplier integration	Supplier integrated product development (turning to suppliers for innovation early on in the concept development phase, bringing in potential suppliers early into the design process)

Table 32 – Continued: Prime-to-Supplier Knowledge Integration

Conceptual Labels	Strategies / Practices	Mechanisms
Information, know-how and skills sharing by network integration	Tight partnerships	Broker supplier marriages (combine complimentary skills for greater benefit of program)
		Mergers with suppliers/competitors (integrates new competencies/skills, improves coordination)
		Long term agreements with suppliers (fosters trust, learning, open sharing)
		Toyota model of vendor village (e.g. developing a common understanding of products, capabilities)

Table 33: Knowledge Integration along Other Channels (Not Included in Conceptual Framework)

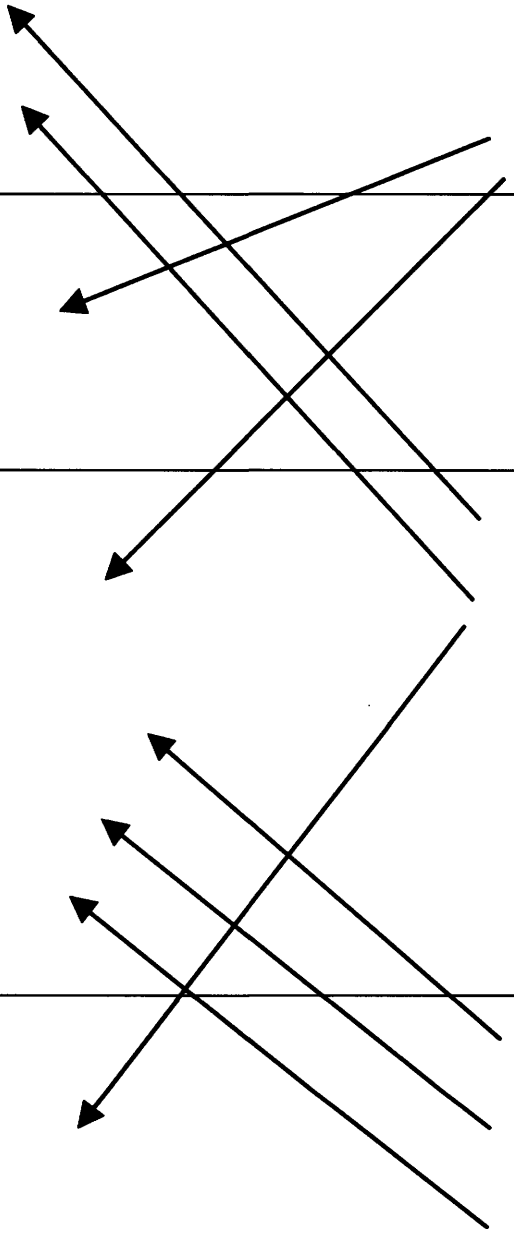
Channels	Conceptual Labels	Strategies / Practices	Mechanisms
Corporate-to-Program	Information sharing by boundary objects	Independent research and development (IRAD)	Transfer of new prime and supplier research discoveries to programs
	Information sharing by organizational coordination	Organizational structuring	Program-oriented organizational structure to integrate across functional disciplines
			Function-oriented organizational structure to integrate across programs
	Information sharing by strategic integration	Process improvement	Program management council (forum for capturing/sharing strategic/higher-order best practices and lessons learned)
	Information sharing by group communication	Routine team reviews	Monthly program reviews by front office (actions to share lessons learned from the top)
Customer-to-Prime	Information sharing by mediation	Customer and supplier management	Program and product team leadership mediate between customer and supplier
	Information sharing by group communication	Routine team meetings	Communication by function groups to transfer feedback to/from customer
	Assistance by group collaboration	Non-routine team meetings	Requirements brainstorming meetings with customer
	Assistance by special review teams	Non-routine team reviews	Independent review team by customer to make technical recommendations
	Assistance by special action teams	Joint problem solving by non-routine team meetings	Taskforce participation by customer experts
Industry-to-Prime	Assistance by special action teams	Joint problem solving by non-routine team meetings	Taskforce participation by academia, industry consultants/experts
	Information sharing by boundary objects	Direct design, technology and process transfer	Systems engineering monitor/roadmap/transfer industry innovations (e.g. commercial off-the-shelf systems (COTS), commercial standards)

Table 33 – Continued: Knowledge Integration along Other Channels

Channels	Conceptual Labels	Strategies / Practices	Mechanisms
Supplier-to-Supplier	Information sharing by boundary objects	Direct design, technology and process transfer	Transfer of design information, buyer furnished equipment (BFE)
	Information sharing by group communication	Routine team meetings	Meetings, teleconferences
			Supplier working groups (e.g. interface contractor working group to communicate interface specification information between suppliers)
	Assistance by mediation	Joint problem solving with boundary spanners	Prime mediated/facilitated collaboration between suppliers to solve common problems

From the concept labeling step presented in the tables above it is possible to extract the following conceptual *categories* and their corresponding *subcategories* for knowledge integration, as well as the relationships between the two, as mapped in Table 34 below:

Table 34: Conceptual Categories and Subcategories for Knowledge Integration

Categories	Relationship of Subcategory to Category	Subcategories
Advice sharing (suggestions for problem solving, typically not binding)		Special review teams Social networking Experts
Assistance (direct involvement in problem solving)		Special action teams Group collaboration Boundary spanners
Information transfer and sharing (explicit knowledge for problem solving)		Personal communication Group communication Boundary objects IT infrastructure Commonality and Standardization Organizational coordination
Know-how and skills sharing (tacit knowledge for problem solving)		Network coordination Training Professional networking

Per the guidelines of the grounded theory method, the four conceptual categories listed in the leftmost column of Table 34 are a "higher-order, more abstract" aggregation of concepts identified in open coding that are similar or related to each other, while the subcategories listed in the rightmost column of the table are an aggregation of similar or related "action/interaction strategies" corresponding to each category (Strauss and Corbin 1990). As such, the subcategories themselves are an abstraction of the actual strategies and mechanisms by which a concept is made operational, and are therefore listed across from the corresponding concept in the previous table. The arrows in the table indicate a "one-to-many" relationship between some subcategories and two or more of the four main conceptual categories. This means that the strategies or mechanisms that constitute the subcategory in question correspond to more than one concept of knowledge integration. For example, mechanisms in the form of experts, such as "tech fellows", integrate knowledge most frequently through giving advice (e.g. making technical suggestions during problem solving), however they also share information across programs in the form of lessons learned and they provide assistance to product teams through direct involvement in the troubleshooting of problems. Therefore, the subcategory labeled "Experts" which is classified under the category "Advice sharing" is also linked with two arrows to the categories labeled "Assistance" and "Information transfer and sharing".

A first examination of the conceptual categories developed in the open coding step shows that the underlying concepts, which are derived directly from the field observations, are in close agreement with the theoretical definition for knowledge integration as proposed in § 2.1.2 of this thesis; that is, the conceptual categories derived from the data embody the transfer and sharing stages of the knowledge integration process through the transfer and sharing of advice, information, know-how and skills, as well as the application of knowledge through direct assistance. This means that the conceptualization of the knowledge integration process using the data from the field research coincides with the earlier conceptualization using insights from existing theory, thus adding to the reliability of the research design (Robson 2002).

The primary outcome of the open coding procedure of the grounded theory method is in the ability to visualize the data at a higher level of abstraction and therefore to better understand the most important aspects and drivers of the phenomenon under investigation. This is illustrated in

Figures 26 and 27 below for routine and non-routine problem solving, respectively, and which reformulate the 'laundry list' of disconnected strategies/practices/mechanisms listed in Tables 27 and 28 above, into a coherent set of concepts for knowledge integration with each concept having its corresponding categories of strategies/practices and the mechanisms through which they are implemented. For example, the conceptual label in Figure 26 about "info sharing by group communication" refers to the concept of "information sharing" through the category of strategies/practices and mechanisms related to the use of group communication.

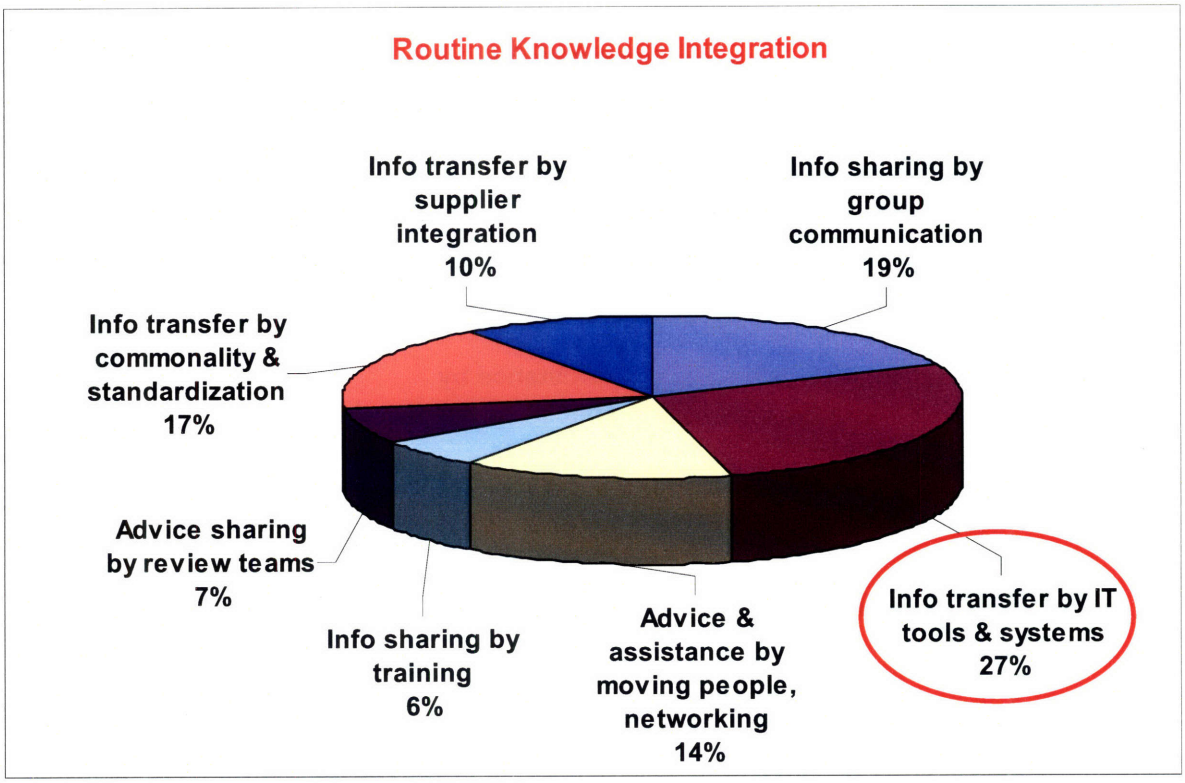


Figure 26: Knowledge Integration in Routine Problem Solving Contexts

As the circled label in the above figure shows, knowledge integration in routine problem solving contexts is dominated by information transfer through IT tools and systems, which empirically validates existing theory (Grant 1996a). Note that for the purposes of enhancing the graph's visibility, some conceptual labels have been folded together such as merging "commonality" and "standardization" together and including "boundary objects" with "IT infrastructure" under "IT

tools and systems". Figure 27 shows the main concepts for knowledge integration in non-routine problem solving contexts:

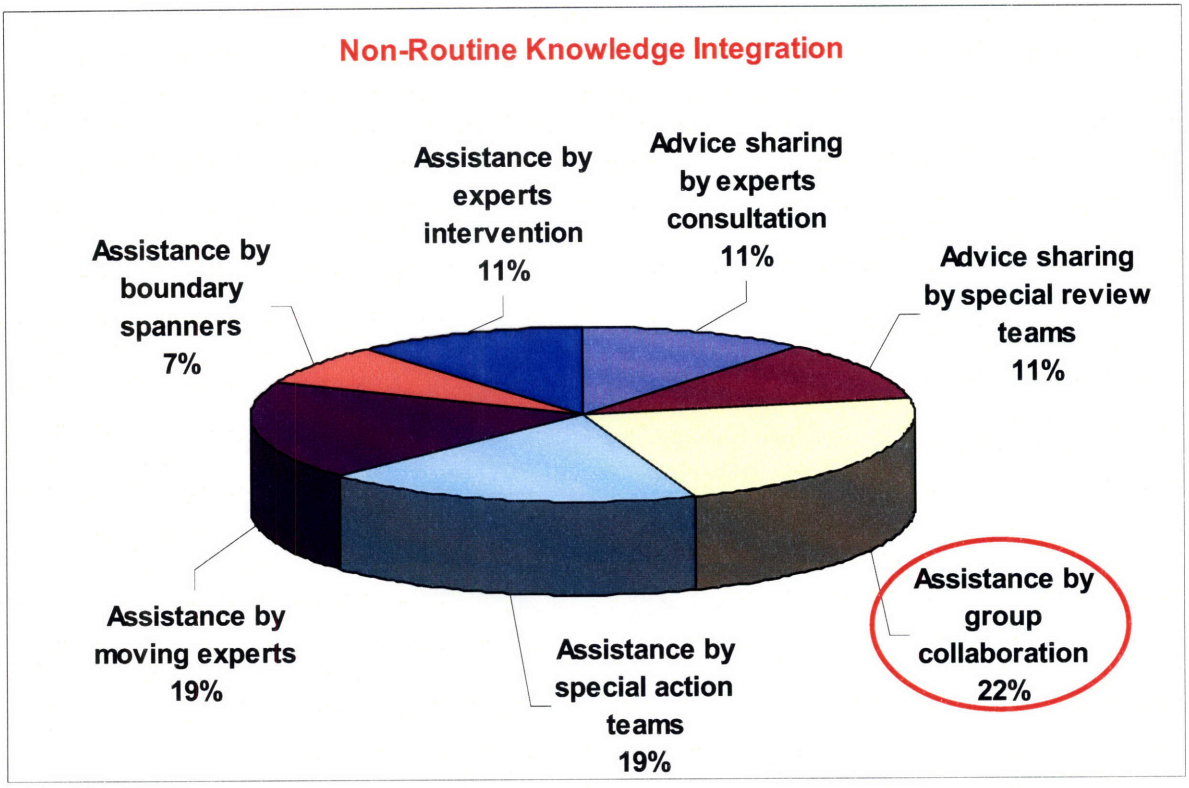


Figure 27: Knowledge Integration in Non-Routine Problem Solving Contexts

The circled label in the above figure empirically validates existing insights about the dominance of group problem solving for integrating knowledge in non-routine problem solving contexts (Grant 1996a).

Another outcome of the coding procedure is the discovery of important *properties* (Strauss and Corbin 1990) pertaining to the developed categories and subcategories and which further characterize the knowledge integration process by adding precision to the developed concepts. For example, some knowledge integration strategies may share a common property of being "routine" even though they are distinct strategies and cannot be grouped together under the same conceptual category, such as "standardization" versus "job rotation". Conversely, similar

strategies may exhibit different properties that should be accounted for in theory development, such as "direct" versus "indirect" transfer of technologies (e.g. directly between programs versus indirectly through a common supplier). Properties thus add further context to the derived concepts and conceptual categories and therefore aid in the development and refinement of the theoretical framework. Properties are also grounded in the data since they are either explicitly mentioned by interviewees (for example when a respondent describes a meeting as "formal") or they are directly inferred from the data (for example when deducing that "walk and talk" is an informal mechanism). These properties emerged in the process of labeling concepts during open coding, as can be seen in Tables 29 through 33 in this section (specifically in the second column of each table under "strategies/practices").

The main properties of interest are: direct and indirect, formal and informal, tacit and explicit, component and architectural, vertical and horizontal, routine and non-routine[38]. These were determined in part based on their frequency of occurrence relative to other possible properties (for example the property "formal" was explicitly cited 24 times in the interviews whereas "long-term" was cited only once, as shown in Table 48 of Appendix B – this served as general guidance for which properties and concepts to take note of in open coding the interview data). Note that these derived properties are also in close agreement with the most notable attributes synthesized from the literature in Table 4 of § 2.1.12, meaning that some of the same properties prescribed in the literature were re-discovered in the coding of the field observations.

Figures 28 and 29 below illustrate the results of coding the interview data using the properties derived from both the literature and the field research for routine and non-routine knowledge integration, respectively. The software tool MAXQDA 2007 for qualitative data analysis was used to assist in the coding of the data in order to manage the scale and complexity of relationships. The results presented in the figures below are based on the coding of a total of 1,349 text segments from all interviews across all three case studies for the routine case, and 264 text segments for the non-routine case (for a total of 1,613 codes). The horizontal axis lists the three different case studies including the prime and two supplier organizations, with the case

[38] Note that the routine/non-routine dimension is already accounted for in the framing of knowledge integration along two separate modes, and therefore does not need to be considered again in the analysis of major properties.

study involving the prime organization being divided into four separate columns to account for each of the four separate rounds of interviews. This was done in order to reflect the changes in the collected data after every refinement of the interview questions[39]. The vertical axis lists the main properties discovered during the coding process. The dots in each figure are a graphical representation of the number of times a particular property was cited in the interviews in a particular case study or round of interviews. A larger size dot means the property in question was more frequently cited.

[39] In contrast, the supplier data which were also collected over three separate rounds of interviews at each supplier organization, were aggregated in the last two columns of Figures 24 and 25 since the field research with the supplier organizations was conducted after the end of the last data collection with the prime organization, and as such there was little to no further refinement of the interview questions at that stage of the research.

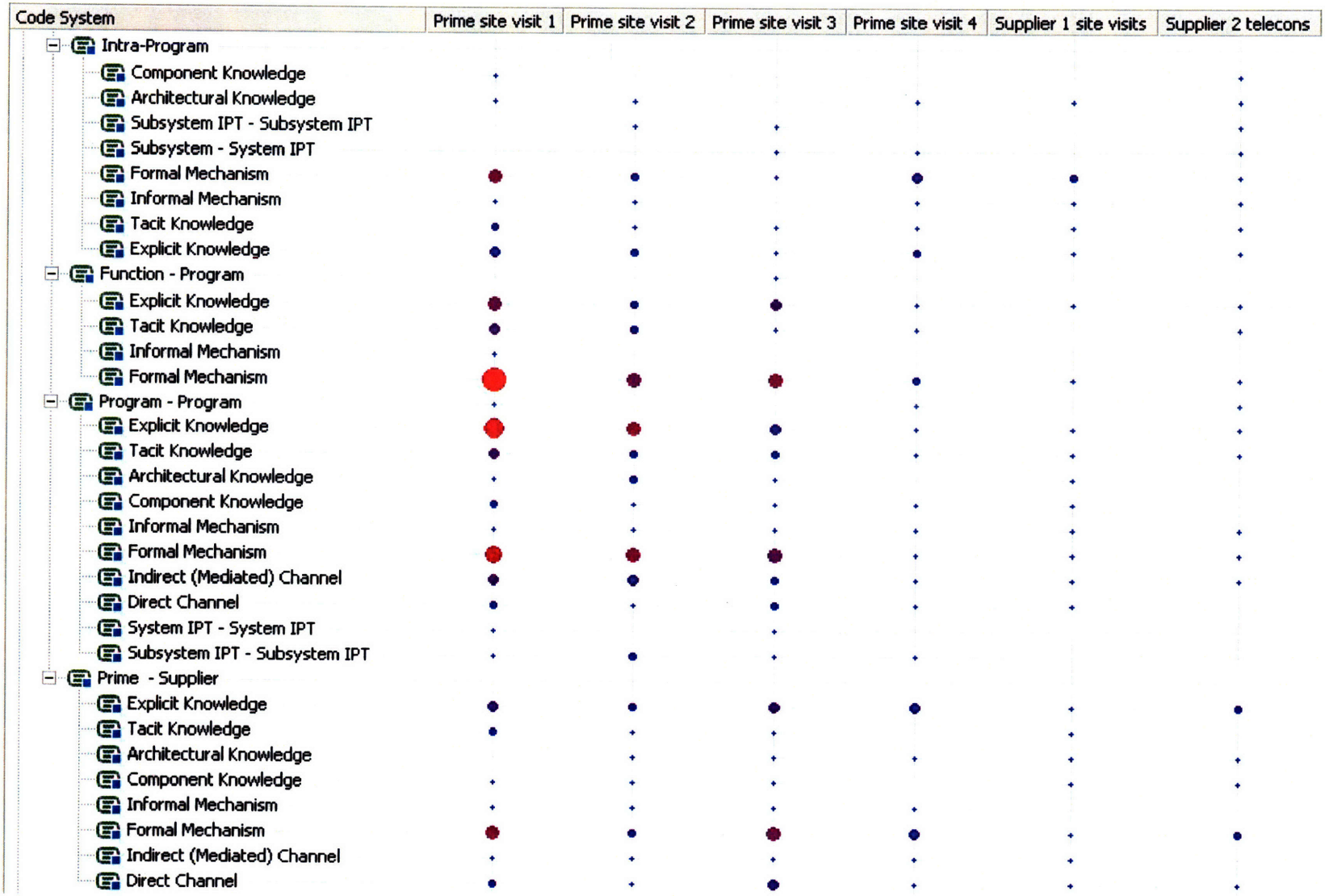


Figure 28: Frequency of Codes for Knowledge Integration Properties in Routine Problem Solving

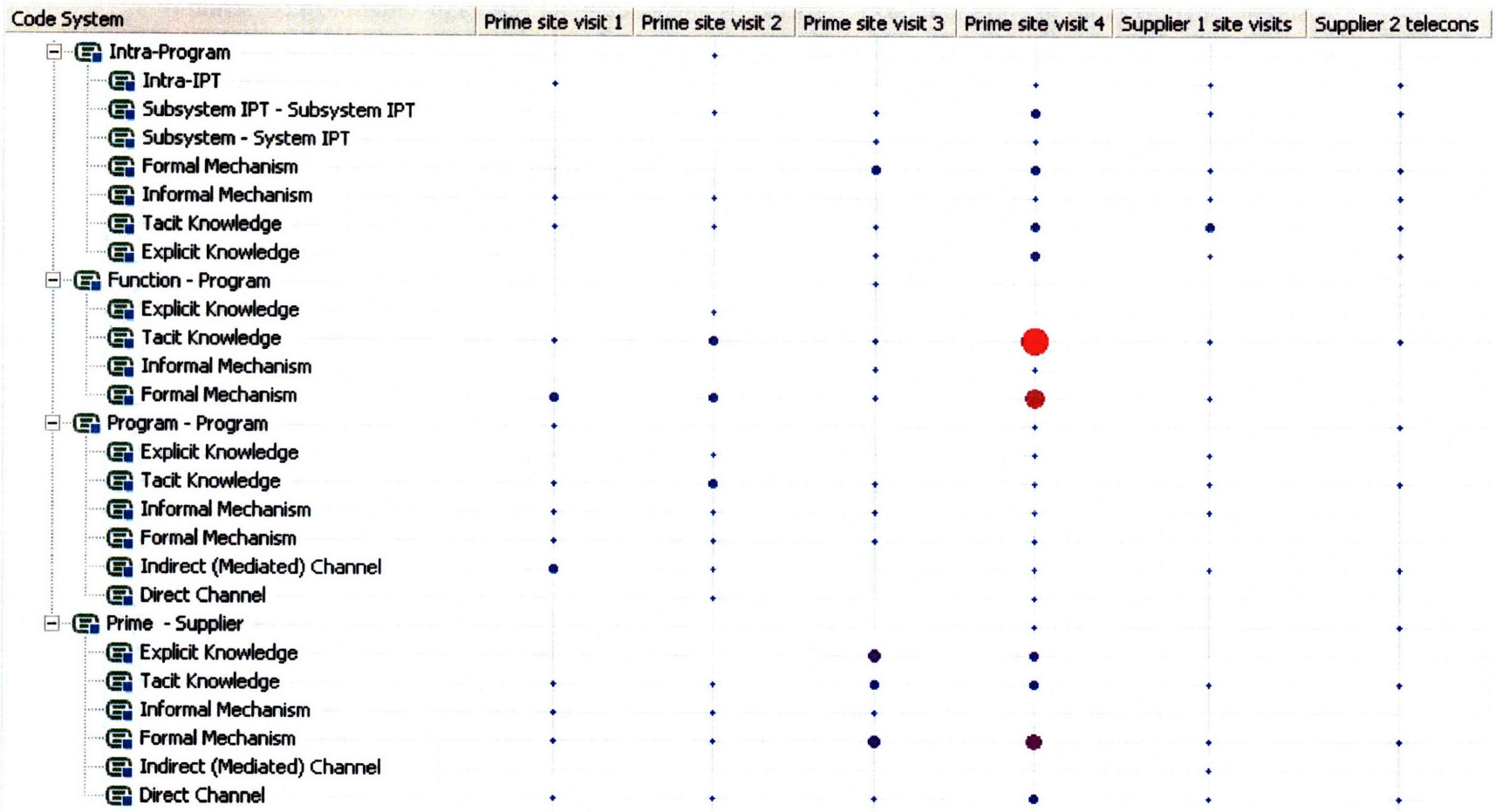


Figure 29: Frequency of Codes for Knowledge Integration Properties in Non-Routine Problem Solving

Note that the first column labeled "Prime site visit 1" represents the first exploratory round of interviews when the interview questions were unstructured and general in nature, so that the resulting insights were mostly related to the "big picture" of knowledge integration and did not touch upon any specific problem solving situations. As such, there were more insights related to routine knowledge integration from that first round than all the subsequent rounds of field research, as reflected in the higher frequency of codes shown in the first of the two figures above (Figure 28). In contrast, the fourth column labeled "Prime site visit 4" represents the last round of interviews with the prime organization during which the structured questionnaire was administered in order to collect data about knowledge integration for specific problem solving events (see § 5.1 for a detailed explanation of the different stages of the field research), therefore the insights from this part of the case study are mostly related to the non-routine integration of knowledge during troubleshooting events. This is reflected by a higher frequency of codes appearing in that column in the non-routine case as shown in the second of the two figures above (Figure 29).

The coding results show that formal strategies, practices and mechanisms dominate the integration of knowledge in general in both the routine and non-routine cases, and especially along the channel linking programs to functions. This is because in project-oriented structures, functions are tasked by design with the formalization of knowledge processes across different programs, such as the standardization of processes in routine work and the deployment of experts in non-routine problem solving. At first glance this would seem to contradict the assertions of many interviewees about the wide-spread use of informal mechanisms, such as in the following direct quotes: "informal exchanges based on personal social networks are the most prevalent form", and "knowledge sharing is mostly informal in walk and talk with colleagues". However, one of the reasons for the discrepancy between the overall data and some specific responses is that informal mechanisms tend to be forgotten by the majority of interviewees since, as one respondent put it, "informal interactions usually precede formal exchanges and serve mostly to establish interest and trust between parties", and therefore they are not always reported in the interviews. In addition, informal mechanisms are typically viewed as ineffective in traditional organizational contexts where following systematic processes is the norm, and as a result they are relatively ignored in the interview responses compared to formal mechanisms. This is

evident in the frequent mention of the "lack of formalization" as a negative aspect and a barrier to knowledge integration, as shown in 5 of the 56 barriers listed in Table 47 of Appendix B.

Examining the frequency of occurrence of knowledge-specific properties, it is clear that explicit knowledge is dominant in routine interactions across all channels, and especially across programs, whereas tacit knowledge interactions become the most prominent during non-routine problem solving, especially across the function-to-program channel. This is in line with insights from the literature that confronting conditions of uncertainty and equivocality, such as those encountered in live troubleshooting, requires richer media (e.g. face-to-face interactions), while routine well-understood tasks can be efficiently accomplished by information systems (Daft and Lengel 1986). There was, however, an exception to the general trend highlighted above in that even in non-routine problem solving, the integration of explicit knowledge was found to be significant along the prime-supplier channel. This can be attributed to the highly formalized nature of interactions between different organizations where everything has to be written down to make it official, but it is also due in large part to the "arm's length" nature of relationships between prime and supplier organizations in this context, as demonstrated by the large number of restrictions and barriers listed in Table 47 of Appendix B. This has the effect of hindering integrated problem solving across organizational boundaries, which is of primary importance in complex product development (Takeishi 2001; Fujimoto 2002). Examining the type of engineering knowledge being integrated (i.e. component versus architectural knowledge) did not provide further insights about the integration process.

Another critical property for knowledge integration is whether the relationship between parties is direct or indirect. As previously mentioned in the literature, direct channels are essential for the integration of tacit knowledge whereas indirect channels are useful for locating new knowledge. The results of the coding process show that external interactions between prime and supplier are direct in nature while the internal interactions across programs are overwhelmingly indirect. This is because the relationship between programs is mostly mediated by the functional organization. Combining these results with the preceding insights about the dominance of explicit knowledge interactions across programs, it is apparent that mediation by functional groups is not sufficient enough to promote the flow of tacit knowledge across program

boundaries. And since tacit knowledge is critical in non-routine problem solving situations, the lack of tacit knowledge flow amounts to a hindrance against the effective management of design interdependencies across boundaries and therefore impedes cross-platform commonality, which is key for efficiency in large-scale complex product development as shown in previous research (Cusumano and Nobeoka 1998). This realization was confirmed by straight quotes from some interviewees that cross-program knowledge integration through the functional organization is inefficient as "a lot of money has been spent, but it is not true integration", especially that there is also a multi-directional pull on functions to serve the needs of multiple programs equally. Finally, coding the interview data for directional properties where vertical integration is represented by subsystem-to-system IPT interactions and horizontal integration is represented by subsystem-to-subsystem IPT interactions within the same program, there was little evidence of any patterns in the data along either dimension, spare one notable indication of an increase in interactions along the horizontal dimension under non-routine conditions. The reasons for this are not clear from the interview data but will be investigated further in the analysis of the questionnaire data for the non-routine problem solving context in § 6.2.

6.1.3 Refined Conceptual Framework for Knowledge Integration

Given the previous analysis of the results of the open coding procedure as discussed in § 6.1.2, and in order to expand on the knowledge integration framework proposed earlier, it is necessary to finally connect the dots between the different conceptual categories by relating them through their subcategories and their corresponding properties. This procedure is known as "axial coding" in the grounded theory literature (Strauss and Corbin 1990; Robson 2002). The dominant concepts related to routine problem solving conditions that emerge from this procedure are shown in Table 35:

Table 35: Knowledge Integration Concepts for Routine Problem Solving

Inta-program	Program-Program	Function-Program	Prime-Supplier
Informal advice sharing		Formal advice sharing	
Direct assistance	Indirect assistance	Direct assistance	
Formal information transfer and sharing			

Similarly, for non-routine problem solving conditions, the dominant concepts that emerge from the axial coding procedure are shown in Table 36:

Table 36: Knowledge Integration Concepts for Non-Routine Problem Solving

Inta-program	Program-Program	Function-Program	Prime-Supplier
Formal advice sharing	--	Formal advice sharing	--
Direct Assistance			

The outcome of the interview analysis is a first-order refinement of the conceptual framework proposed in Chapter 3 of this thesis for both routine and non-routine problem solving, as shown in Figure 30 below:

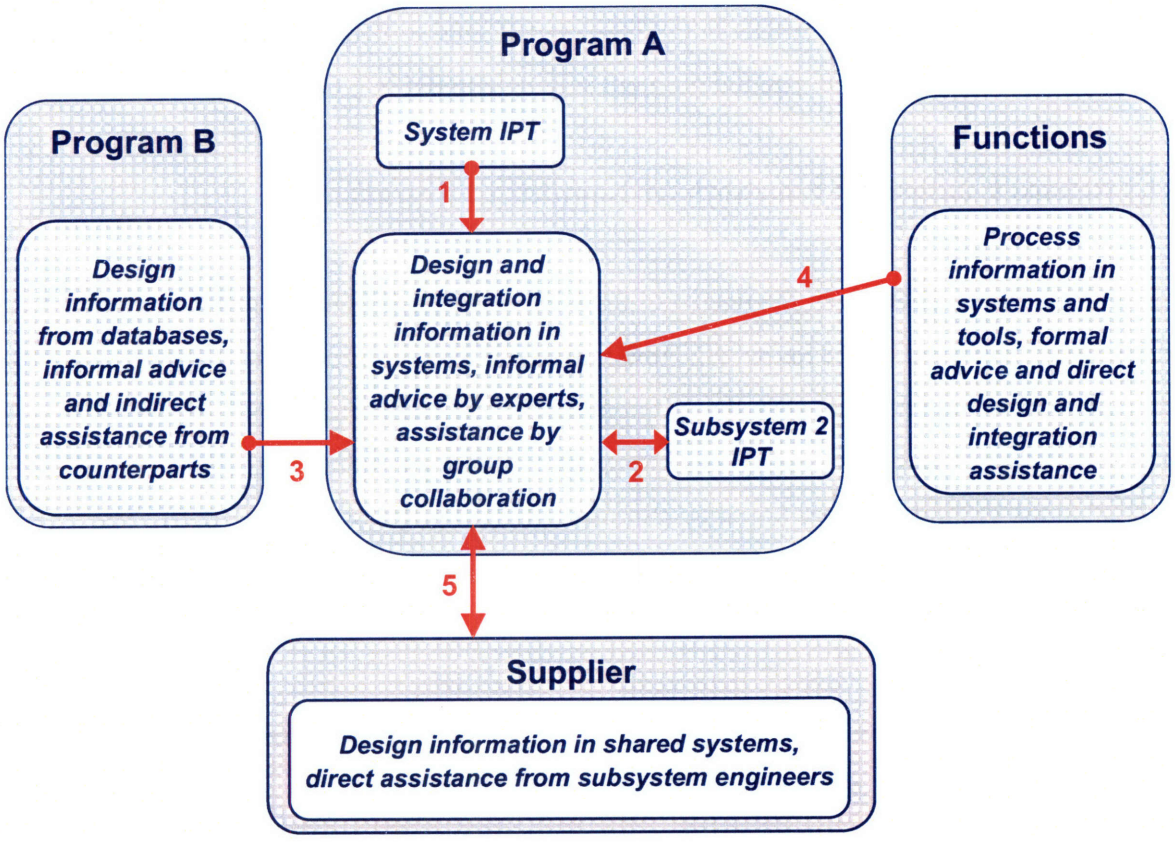


Figure 30: Refined Knowledge Integration Framework

The above framework presents a comprehensive picture of the knowledge integration process in large-scale complex product development. However, the picture remains static in that there is no visualization of the dynamics of knowledge integration under different non-routine conditions. In other words, since non-routine problem solving drives the use of different channels, strategies and mechanisms for integrating knowledge, the static framework by itself cannot completely describe the integration process and a more dynamic picture of knowledge integration is needed to accurately describe the phenomenon in its complex setting. It is thus necessary to investigate the non-routine problem solving context in more detail in order to answer the questions:

- What channels and mechanisms are used in different problem solving contexts; and,
- Which ones are used most frequently?

Answering these questions requires a more structured investigation than can be accomplished using interviews, and therefore a structured questionnaire instrument was administered across all case studies in order to collect categorical data about the use of knowledge integration channels and mechanisms for a wide variety of major problem solving events, as will be explored in detail in § 6.2 below.

6.2 Quantitative Analysis of the Questionnaire Data

The problem solving and knowledge integration data collected with the structured questionnaire instrument are presented in Tables 49 to 51 of Appendix C. There are a total of 49 problem solving cases collected with the structured questionnaire administered across all case studies in the final round of interviews and focusing on four major avionics systems, namely the Communication Navigation Identification (CNI) system, the Electronic Warfare (EW) system, the Mission Computer (MC) and the Multi-Function Radar. In addition, and as part of the 49 cases, there are a number of problems involving aircraft systems outside the avionics suite, including several mechanical, structural, electronic and display systems. The analysis in this section uses the different system and problem characteristics identified in the cases to frame the dynamics of knowledge integration under different problem solving conditions.

6.2.1 The Influence of Product Architecture on Knowledge Integration

The type of architecture for the system in which the problem originated was collected by asking respondents to choose between the two opposite forms: integral or modular. The reported choice was then validated where possible against publicly available sources in the literature for the system in question. Another check was done by estimating the *degree of modularity*[40] of each system based on the *average spread* of problems attributed to that particular system. This estimation was based on the assumption that problems occurring in integrated systems ripple across several other systems due to the tight coupling between them, whereas problems occurring in modular systems tend to be more self-contained or affect only a few other systems due to the loose coupling between them. This assumption builds on existing insights in the literature about the properties of loosely coupled versus tightly coupled architectures, and was further validated by several practitioners in the final round of field interviews.

As an example, there are 6 problem cases attributed to the EW system, where 2 problems remained internal to the system itself, 1 problem affected 1 other system, and 3 problems rippled across 5 other systems each[41]. On average, the spread of EW problems to other systems is equal to the average number of all affected systems (i.e. the average of 1+1+2+5+5 = 3.17 affected systems on average), which when normalized from the measured "1-to-5" scale to a "0-to-1" scale where 0 is a purely integral architecture and 1 is purely modular, gives a degree of modularity of 0.46. This number is less than 0.5 and therefore the EW system architecture is considered to be integral. This calculation corroborates the responses provided by EW system experts in the field and is supported by the manufacturer's information and other literature about the subject EW systems for the aircraft programs surveyed. The same calculation was done to verify the system architectures of all other avionics and non-avionics systems in all 49 problem cases.

[40] Note that there are already several methods for measuring the degree of modularity of a complex system (Ulrich 1995, Sosa, Eppinger, et al. 2003, Hölttä, Suh et al. 2005), however they all require deep knowledge of the internal workings of the system (e.g. the number of system components and functions, the functional interactions between components), which was not available in this research due to classified and proprietary restrictions.

[41] Recall from § 5.2 that a score of 1 to 5 was used to measure the spillover of a problem in one system into other systems, so that a maximum score of 5 was used even if a problem affected more than 5 other systems. This was done to manage the difficulty of obtaining actual numbers of affected systems for far-reaching problems.

By the above classification and as shown in Table 49 of Appendix C, there are 26 problem cases pertaining to systems with an integral architecture (namely the Mission Computer system, the EW suite and the Electronic Display systems for all three aircraft case studies, as well as the CNI system and 3 other non-avionics systems in case study "C" only). The remaining 23 problem cases originated in systems with a modular architecture (namely the Radar system and 9 other non-avionics systems in all three case studies, as well the CNI system in case studies "A" and "B" only)[42]. The corresponding knowledge integration data are plotted in Figures 31 and 32 for integral and modular cases, respectively. The figures illustrate the frequency of use for each of the 5 knowledge integration channels, as framed in the initial theoretical framework proposed in Chapter 3 of this thesis, in each of the two system architecture regimes (e.g. channel #2 circled below is used in 84.6% of all problem solving cases pertaining to integrated systems only).

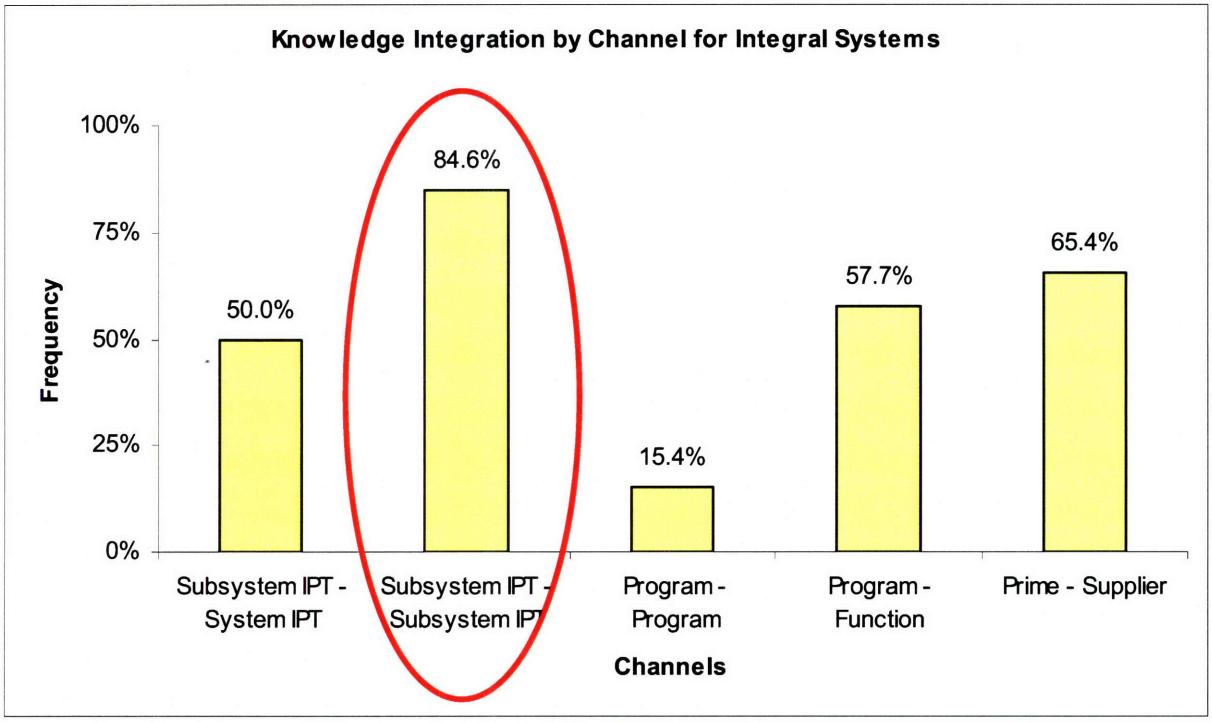


Figure 31: Knowledge Integration for Problem Solving in Integrated Architectures

[42] For the purpose of the following analysis, recall that troubleshooting modular systems involves debugging module interfacing problems, whereas troubleshooting integral systems involves tuning multiple parts of the system due to part interdependence (Ulrich, 1995).

As illustrated for the integral architecture case, the most frequently used channel for integrating knowledge in non-routine problem solving involving integrated systems is the intra-program channel between different subsystem IPT's, while the least used channel in this case is the program-to-program channel. First, the high frequency of intra-program interactions between subsystem IPT's is because of the tightly coupled nature of integral systems where a problem in one subsystem affects other parts of the system, such that the subsystem where the problem manifests itself may not be where the source of the problem is located or "rooted". In other words, integration creates ambiguity in problem solving which necessitates frequent back-and-forth knowledge interactions between the concerned IPT's in order to diagnose the problem and develop a solution[43]. Along those lines, an important observation from this research is that iterative problem solving between subsystem IPT's can in many cases turn to "finger-pointing" or "throwing-over-the-wall" type of interactions between IPT's, which when combined with each team's self-sufficient nature can lead to problem solving in isolation. What this means is that focusing on the frequency of interactions exclusively can be a misleading indication of close collaboration when the reality is completely opposite. This is because the effectiveness of the knowledge integration process also depends on the richness of interactions, not just their frequency. This is why it is also important to evaluate the actual mechanisms for accomplishing knowledge integration along each channel, and this research has found that 68% of all mechanisms used along the subsystem-to-subsystem IPT channel in integrated architectures are information systems for the exchange of explicit knowledge alone (e.g. common problem reporting system, integrated tools). These impersonal interactions are therefore one possible explanation for the non-collaborative behavior between IPT's, and it was also found in this research that the over-reliance on information systems in this context is a potential source for requirements creep since as one interviewee put it, anyone can file a common problem report (CPR) through these systems without having to go through rich discussions to assert whether the observed anomaly is real or a "false positive". Therefore there is a clear indication from this research that root-cause analysis in integrated system architectures can benefit from tacit knowledge interactions to supplement the over-reliance on information systems in this process.

[43] This process is known as "root-cause analysis" or "failure analysis" in which various methods and tools are employed to locate the 'culprit' part of the system and to develop the appropriate solution. In complex integrated systems, root-cause analysis is a very systematic and systemic process which requires the involvement of multiple teams responsible for different parts of the system.

Key takeaway:

In highly integrated systems, rich tacit knowledge integration between IPT's responsible for different subsystems is critical for efficient and effective root-cause analysis

Second, and in terms of knowledge flow between programs, the primary reason for the low frequency of interactions along the program-to-program channel is that integration and customization in systems go hand in hand, with both strategies being a means to increase performance and both having a self-reinforcing relationship, and as such the more highly integrated the system, the more customized it is and the less knowledge commonality there is with other systems of the same type in other programs. This reduces the usefulness of cross-program interactions as reflected in the figure above. This is in contrast to modular systems which have significant commonality between them, as discussed for Figure 32 below.

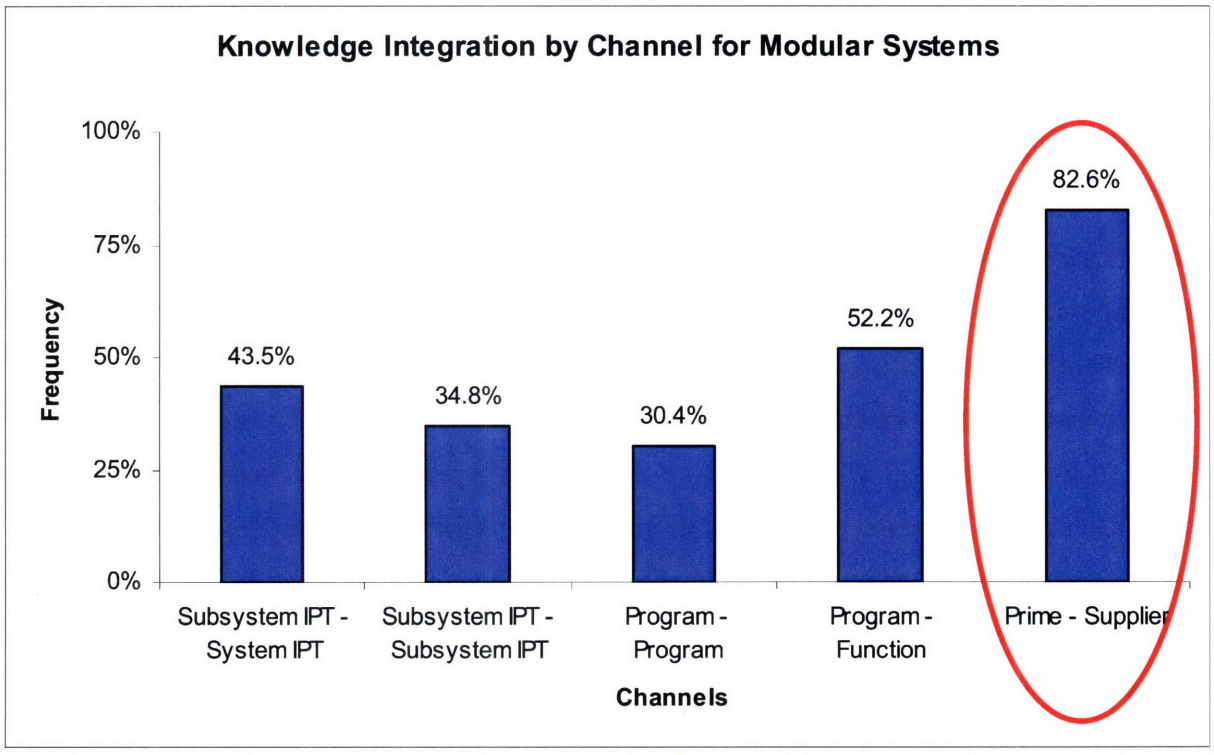


Figure 32: Knowledge Integration for Problem Solving in Integrated Architectures

In the modular case illustrated here, the most used channel for integrating knowledge is across the prime-supplier boundary, while the least used channel is across the program-to-program interface. One obvious explanation for the high frequency of prime-supplier interactions is that in complex product development, subsystem modules are typically designed and developed by external suppliers who retain most of the knowledge about the internal workings of their own modules. Therefore problem solving in a modular architecture has to go through the supplier of the module where the problem is rooted or where the problem manifests itself. This would appear to validate the common assumption that system integrators in the prime organization do not have the necessary design knowledge to troubleshoot problems with the supplier's box. However, this explanation alone does not constitute the full story, since subsystem design problems account for only 24.5 % of all 49 problem cases collected, while integration-type problems account for 57.1 % of all problem cases. As such, a majority of problems should not in theory require frequent supplier involvement, since it is commonly accepted that system integration knowledge is closely held by the prime system integrator only and not by the suppliers of the affected modules. The contradiction between observation and accepted theory opens the door for alternative explanations such as the fact that modular architectures are used by the prime organization to "hide" its architectural knowledge from its supplier (Sanchez and Mahoney 1996; Baldwin and Clark 1997), thus eventually leading to system integration problems and forcing both prime and supplier to exchange knowledge during problem solving. This argument is well supported in the field interviews where the prime-supplier relationship was often characterized as being at "arm's length" instead of a close partnership, for example in the following quote: "We need to move beyond the 'Purchase Order' mentality and treat our suppliers as partners instead of vendors". And in fact, this research has found that 69% of all mechanisms used along the prime-supplier channel in problem solving under modular architectures were for the integration of explicit knowledge (e.g. requirements and specifications changes and interface control documents, engineering change documents). This empirically confirms existing insights about the nature of prime-supplier interactions being mostly a distant and formal buyer-vendor relationship. However, before accepting these observations as a full explanation for the multi-faceted relationship between prime and supplier organizations, it is necessary to further explore the knowledge integration process along this channel for different

problem solving contexts (specifically for design versus integration problems). This will be addressed in § 6.2.2.

Key takeaways:

1) The high frequency of knowledge interactions between prime and supplier organizations in modular architecture regimes is not restricted to design problems

2) Problem solving in modular systems can benefit from richer tacit knowledge interactions between prime and supplier organizations

When examining the low knowledge integration activity along the program-to-program channel, it appears at first that the outcome is contrary to expectations, since modular systems typically have a high degree of commonality with similar systems in other programs and can therefore benefit from integrating knowledge about commonly encountered problems across programs. However, as explained already in the qualitative analysis under § 6.1.1, the program-to-program interface was found to be walled-off by a host of barriers ranging from classified and proprietary restrictions to short-sighted strategies of local program optimization and close guarding of knowledge assets. This is compounded by the fact that project structures which are common in complex product development end up isolating programs from each other so that cross-program interactions are only on a need basis. Furthermore, and as already pointed out in § 6.1.1, it was found that some of the most common mechanisms for integrating knowledge across programs are shared databases and common repositories, which suffer from several shortcomings such as information overflow and obsolescence, and are therefore rarely used and ineffective for major problem solving. However, it should be pointed out that compared with the integral case, there is a significant increase (by almost two-folds) in the use of the program-to-program channel in the modular case. This constitutes an indication of frustrated attempts at integrating knowledge from other programs for major problem solving in modular architectures.

A detailed comparison between the integral and the modular cases will be made to determine if the differences in the use of knowledge integration channels and mechanisms between the two cases are significant or due to random error. The comparison is first illustrated in Figure 33 below:

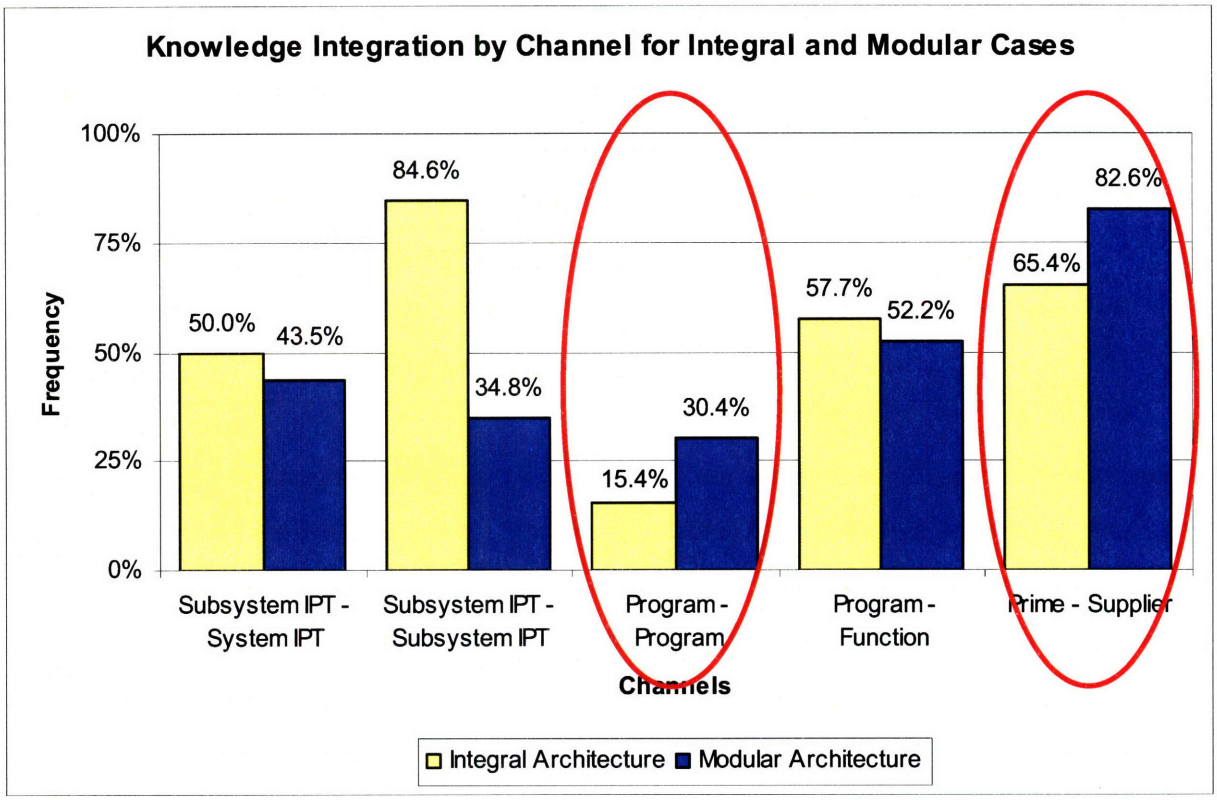


Figure 33: Knowledge Integration in Different Architecture Regimes

Starting with a visual comparison, it is clear that there is little difference (less than 10 %) in the frequency of integration along channel 1 (subsystem IPT-to-system IPT) and channel 4 (program-to-function), so by this simple "eyeball test" it is possible to say that knowledge integration along these two channels is not sensitive to changes in the system architecture (i.e. the use of these two channels for integrating knowledge in order to solve a problem in a particular subsystem is invariant with respect to the type of architecture of that subsystem).

However, for channel 2 (subsystem IPT-subsystem IPT), channel 3 (program-to-program) and channel 5 (prime-to-supplier), there are apparent differences (greater than 10 %) in the frequency of knowledge integration along each channel between the integral and modular cases. In order to determine whether this difference is statistically significant, a chi-square test of independence was performed on the data and a value of $\chi^2 = 7.23$ was calculated, as shown in Appendix D.

This result is statistically significant at the 95 % confidence level[44], and therefore we can accept the alternative hypothesis (Ha) that there is a relationship between system architecture and knowledge integration in complex product development. These outcomes can be depicted in the original framework by emphasizing the channels which are most frequently used in different architecture regimes, as shown in Figure 34 below for integrated architectures where the subsystem-to-subsystem IPT channel is most frequently employed:

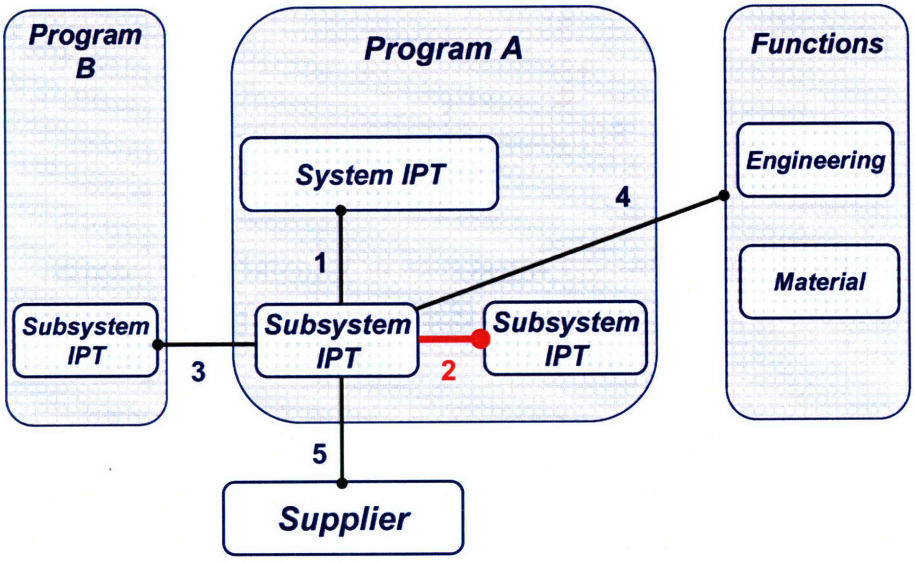


Figure 34: Knowledge Integration in Integrated Architectures (Channel 2 Most Frequently Used)

The implication of this finding for problem solving is that integrated architectures make the problem solving process more iterative and therefore more difficult (note that the problems themselves may not be more difficult because of integration, but the problem solving process becomes increasingly vague and requires more knowledge interactions to resolve the ambiguity). Similarly, Figure 35 below frames the knowledge integration process for modular architectures where the program-to-program and prime-to-supplier channels are most frequently employed:

[44] The calculated chi-square value is greater than the alpha significance level $\alpha = 5.99$ corresponding to p (0.05) for 2 degrees of freedom in the chi-square distribution table; in other words the probability of mistakenly accepting the alternative hypothesis is very small and we can reject the null hypothesis (Ho) that differences in the frequency of knowledge integration along channels 2, 3 and 5 for different system architectures are due to random chance alone.

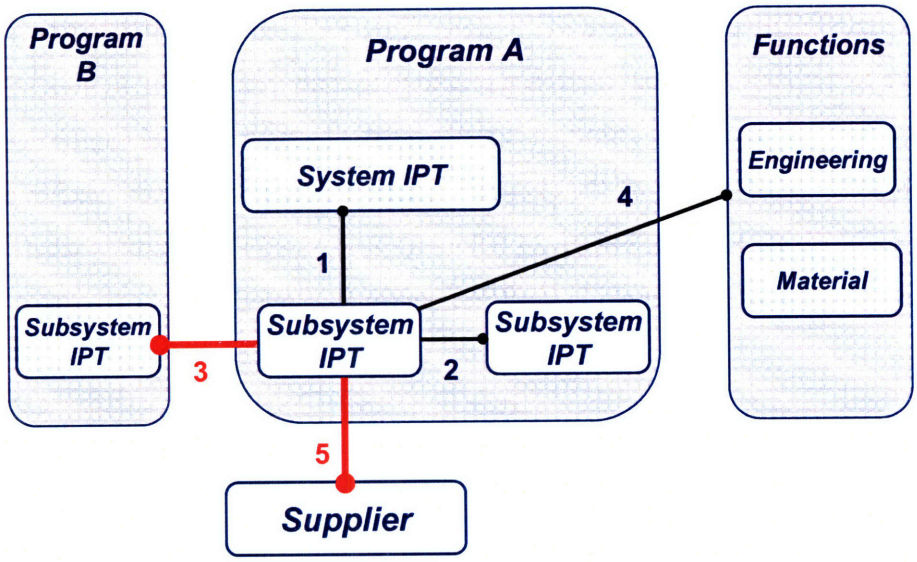


Figure 35: Knowledge Integration in Modular Architectures
(Channels 3 and 5 Most Frequently Used)

The implication of this finding for problem solving is that problem solving in modular architectures relies mostly on prime-supplier knowledge integration, and can potentially benefit in a substantial way from breaking down barriers across programs.

Key takeaway:

Despite the barriers against effective knowledge integration between programs, the knowledge interactions along the program-to-program channel are markedly higher for problem solving in modular architectures than for integral systems

It is also important to note that despite the marked difference in the frequency of prime-supplier interactions for modular versus integrated systems, there is nonetheless a high degree of interactions (over 65 %) in both cases, which for the integral case runs counter to common wisdom (recall the argument outlined in § 2.3.1 of the literature review chapter about the higher efficiency of coordinating knowledge internally to the single firm than across external boundaries when dealing with tightly coupled products (Christensen, Verlinden et al. 1999)). However, as already argued in this thesis the high degree of complexity of the systems in

question makes it impossible for one firm to have all the knowledge required to develop the entire system, thus necessitating frequent interactions with external partners even for integrated systems. In addition, however much a complex system is modularized there will always be a high degree of coupling between the assembled modules in the overall system, which means that problems in one module can lead to integration problems in the entire system, and ultimately to frequent joint problem solving with module suppliers. In the case of military avionics, the ever increasing demand for cutting-edge performance and added capability translates to increasing customization and integration of the avionics suite and the individual systems themselves, which leads to a corresponding increase in overall system complexity. A widely adopted strategy for dealing with complexity in this context is to federate large "chunks" of each system while maintaining a high degree of integration between the different chunks[45] in order to maintain a high degree of performance. This means that there is frequent supplier involvement in troubleshooting for the vast majority of avionics systems, regardless of the degree of integration of each system, as manifested in the data. These frequent prime-supplier interactions as observed in the integral case support previous research findings suggesting that teams with experience in the development of highly integrated systems are better able to communicate technical information across external organizational boundaries (Sosa, Eppinger et al. 2003). Note that a high degree of prime-supplier knowledge integration in the development of complex integrated systems is not necessarily applicable for developing systems which do not exhibit a high level of complexity. In the low complexity case, the degree of knowledge integration with suppliers would be high for modular architectures only.

Key takeaway:

There is a high level of involvement by suppliers in troubleshooting problems with the prime organization regardless of the architecture of the system experiencing the problem. This is due to the increasing degree of complexity in high-technology systems and is an indication of migrating system integration knowledge to the supplier base.

[45] This architecture is colloquially known as "Fedegrated", which is a mix between federated and integrated architectures.

It also follows from the above that in complex systems development, supplier involvement in problem solving is not only limited to design problems alone as typically believed. The following section will explore knowledge integration for different types of problems.

6.2.2 The Influence of Problem Type on Knowledge Integration

The 49 problem solving cases collected through the questionnaire and relating to the system design and development phases (SDD) of product development can be classified along six different categories of problem types, which are: 1) hardware design, 2) software design, 3) system design (for either hardware or software systems), 4) hardware integration (known as packaging problems in avionics), 5) software integration (such as sensor fusion problems), and 6) system integration problems (involving both hardware and software). Some problems are also a mix of either hardware design and hardware integration issues, or software design and software integration issues. For simplicity of the analysis, the problem cases collected in this research are classified along two main categories: design problems and integration problems[46]. Design problems are generally due to anomalies in the subsystem engineering phase of the design process, while integration problems, which typically account for the majority of major problems in complex product development, are generally due to interface incompatibilities (i.e. mismatched interface specifications, bad requirements and different standards). This was confirmed in the field research through several direct quotes from interviewees, for example: "most problems are related to interfaces", and "modules work fine on their own most of the time until they are plugged into the larger assembly". However, it is also the case that some integration problems are rooted in bad design decisions at the subsystem level which spillover across interfaces into other parts of the system, especially in tightly coupled complex architectures such as the avionics systems considered in this research.

Figure 36 shows the differences in the frequency of knowledge integration for design versus integration problems. Note that the graph does not distinguish between different system

[46] Note that there are other common problems in the SDD phases of development such as manufacturing problems, process and procedures issues and failures relating to testing and manufacturing equipment, or even errors in flowing incorrect data and documents. However, the most complex problems at these stages of development are typically design and integration problems.

architectures, only between different types of problems regardless of the architecture of the system experiencing the problem.

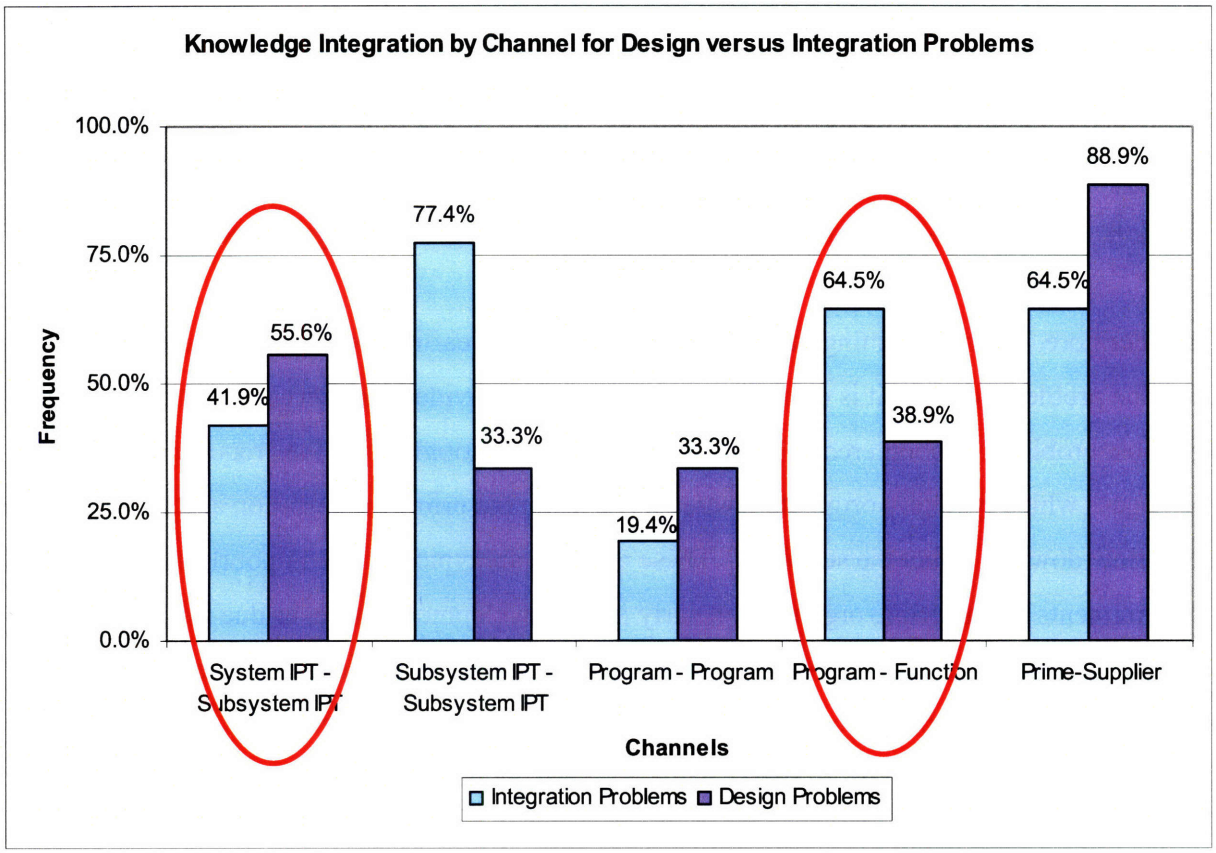


Figure 36: Knowledge Integration for Different Problem Types

As expected, knowledge integration along the prime-supplier channel is very high for subsystem design problems. This could be due to the common reason that design knowledge needed for problem solving is found in the supplier base since suppliers are tasked with the design and development of subsystems in complex product development, as already pointed out in the previous section. However, it could also be the case that suppliers are the source of design problems and that knowledge integration along this channel is due to the prime organization having to help its suppliers with problem solving. Examining the questionnaire data about the different solution approaches to design problems as described by respondents, it is possible to determine where problems originated from and in which direction knowledge flowed to solve the

problem. Hence, from the problem solving narratives, the supplier was determined to be the source of knowledge for problem solving in 58.8 % of all design problems. This result is not conclusive since it is relatively close to an even outcome[47]; however, it is a first indication that the main reason behind the high frequency of knowledge integration along channel #5 is for supplier assistance in solving design problems. In fact, most design problems (87.8%) are estimated to originate at the prime organization. This is also despite the potential for bias by the respondents, the majority of them being affiliated with the prime, as there is greater likelihood for individuals to avoid reporting problems originating at their own organization.

Furthermore, when examining the interview data from the earlier rounds of the field research for routine problem solving, it is clear from several quotes provided by interviewees that the source of most problems encountered in day-to-day product development are the "requirements changes" which typically originate upstream (i.e. from the prime organization and the customer) and flow down to affect the suppliers. These changes are typically due to poorly articulated requirements by the prime organization early in the development phase, or due to changing customer needs over time, or even to the poor flow-down of the correct requirements and specifications by the prime to the suppliers. This is evident from direct quotes by interviewees that "the real source of most problems is the poor articulation of high-level customer requirements" and that "subsystem-level requirements need to be flowed down better". This reality does not seem to be reflected in the non-routine problem solving data above since suppliers are deemed to be responsible for a high 41.2 % of major problems (i.e. 100% – 58.8%). However, since this statistic is for major non-routine problems as surveyed in the 49 problem cases in this research, it is therefore not reflective of the day-to-day reality of product development where routine changes in requirements and specifications upstream account for the majority of problems, and we can conclude that suppliers are not the source of most problems in general. It also follows that while routine problems due to minor requirements changes may be deceptively seen as "small" compared with major engineering changes, their impact can be quite disruptive to the development process due to their repetitive nature and the tediousness of the

[47] Note that the determination of which organization is responsible for developing the solution is a conservative estimate based on the final solution outcome, so that for example if the solution involved a change of requirements by the prime organization, then the prime is credited with the solution, which ignores the potential role of the supplier in problem solving. This is because the data collected through the questionnaire does not allow for estimating the relative role of each organization in developing the solution.

corresponding problem solving process. This is confirmed by several quotes from the interviewees in both the prime and supplier organizations where they assert that "it's the little issues that eat your lunch" and "it's the small things that drag you down, not the big problems". It is therefore more plausible to assume that most design problems (big and small combined) originate in the prime organization, and that overall, the design knowledge for problem solving is largely found in the supplier base.

Examining the figure above for system integration problems, the frequent knowledge interactions along the prime-supplier channel for system integration problems seem to contradict expectations, since system integration knowledge is typically held by the prime system integrator. And despite the fact that these interactions are less frequent than those for design problems as would be expected (i.e. the prime-supplier channel is used 26 % less for integrating knowledge about integration problems than for design problems), they are nonetheless significant enough (almost 65% of the time) that they still warrant further investigation. The most probable explanation can again be obtained from examining the questionnaire data related to the description of problems and the corresponding solution approach. In this case, the prime organization was found to be the source of solutions to integration problems 63.2 % of the time. This is in line with expectations that the prime system integrator holds most of the architectural knowledge; however supplier contributions to problem solving for integration problems remain high (at 100% – 63.2% = 36.8 % of solutions come from suppliers). This means that suppliers are involved in problem solving with the prime regardless of the types of problems encountered (or in other words, not just for subsystem design problems as commonly believed). This confirms the conclusion reached in the previous section that in dealing with increasing complexity through the outsourcing of larger chunks of the overall system, knowledge from the supplier base is increasingly needed for solving problems. Note that several integration problems were found to be caused by design problems rippling across other parts of the system, as expected in the case of tightly coupled complex avionics systems.

For system integration problems, it is clear that most troubleshooting happens between different IPT's responsible for the different subsystems, which is in line with the preceding findings for problem solving in integrated architectures. It is also noteworthy that this same channel is hardly

used when dealing with design problems since it exhibits the lowest frequency of knowledge interactions in this case. This suggests that design problems tend to remain more localized (or internal to one subsystem) and thus do not involve other subsystems, whereas integration problems, which are by definition problems that involve two or more subsystems, require extensive knowledge interactions with other subsystem IPT's. It is also intuitive that integration problems do not require as much knowledge flow across programs as design problems do, which is exhibited in the collected data and illustrated in the figure above. This is because integration problems are more likely to be unique to the interfaces of the subsystems involved (i.e. interface-specific) and therefore similar problems are less likely to be commonly experienced with systems having different architectures in other programs.

Finally, it is noteworthy that knowledge integration along the subsystem-to-system IPT channel is much higher for design problems while the reverse is true for the program-to-function interface. To explain the first part of this empirical result, recall the previous findings from the preceding interview analysis where functional groups were found to be the source of deep engineering expertise and skills as well as scientific knowledge as shown in Figure 25, which makes them an ideal source of troubleshooting knowledge, especially in terms of root-cause analysis know-how and skills necessary for tackling complex system integration problems. In fact, this research found clear empirical evidence of extensive assistance (87% of all knowledge integration mechanisms employed along the program-to-supplier channel) in this context. Conversely, the main reason for the extensive involvement of system-level IPT's in solving design problems with the suppliers are found in the problem solving narratives from the questionnaire where there is evidence of an increasing need for mediation and oversight by system IPT leaders, chief engineers and higher program authority in order to resolve deadlocked problems with the supplier (such as entrenched problems with the supplier's process). Specifically, this involvement was due to the need for mediation by the chief engineer to resolve technical issues which had spilled over into disagreements and mistrust between prime and supplier, or even between customer and supplier, and it was found that 44% of all mechanisms employed along this channel are for sharing advice by higher program authority with suppliers, notably in deadlocked problem situations.

Key takeaways:

1) Functional groups are a key source of expertise, know-how and skills for system integration problems in complex product development

2) Advice sharing by higher program authority with suppliers is a critical mechanism for mediating the resolution of design problems where the prime system integrator is increasingly at the mercy of suppliers for design knowledge

Figures 37 and 38 below frame the knowledge integration process for design and integration problems, respectively:

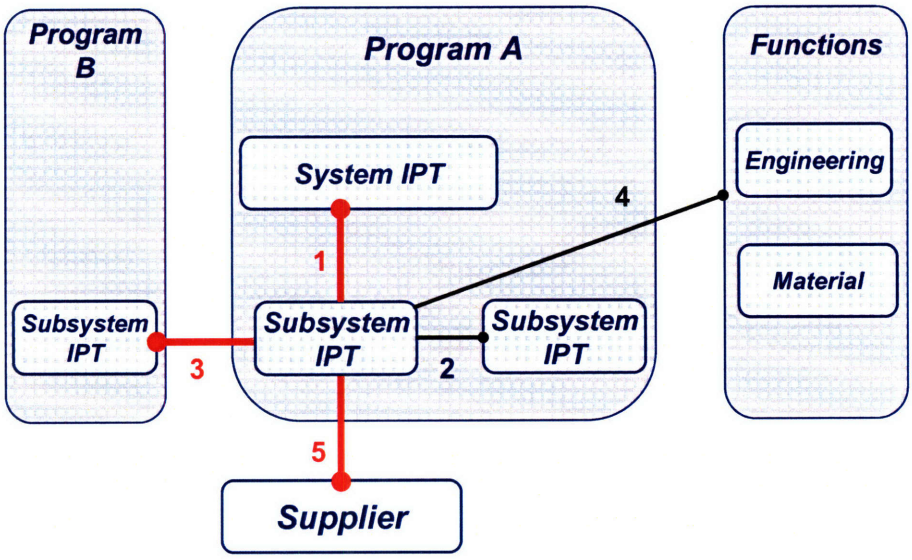


Figure 37: Knowledge Integration for Design Problems
(Channel 5 Most Frequently Used, Followed by Channels 1 and 3)

In contrast to the knowledge integration picture for design problems depicted above, system integration problems require frequent use of the program-to-function channel since most problem-solving is done by the prime organization, as opposed to design problems tackled mostly by the suppliers and where the prime's functional groups are not called upon for help as frequently. This is depicted in Figure 38 below. Note that the knowledge integration process with functions will be explored further in the following sections addressing the influence of

problem novelty and technology maturity on the frequency of knowledge interactions. Also, and as already explained in the previous section, knowledge interactions across programs for system integration problems are more limited than for design problems, due to the fact that these types of problems are unique to each system architecture, and therefore there is little commonality between integration problems across programs. Furthermore, while the data show that knowledge integration along channel #1 (with the overall system IPT) is high for integration problems, insights from the interviews have revealed that this channel is used only when problems ripple across a large number of other systems or when problem solving triggers red flags with respect to budget and schedule.

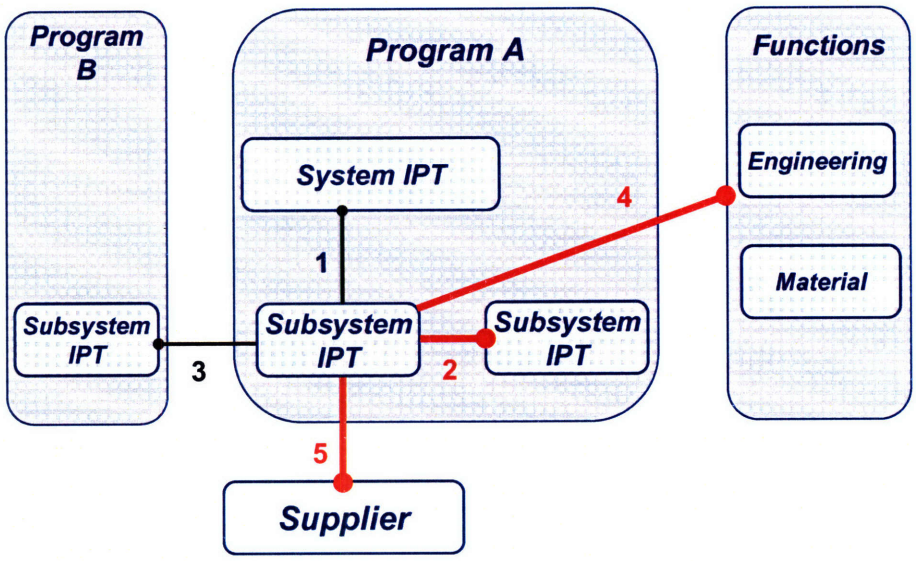


Figure 38: Knowledge Integration for System Integration Problem Solving (Channel 2 Most Frequently Used, Followed by Channels 4 and 5)

Having captured the dynamics of knowledge integration for different system architectures and different problem types, the following two sections will address the influence of problem complexity on knowledge integration through variations in problem novelty and technology maturity. This is important because, as already discussed in chapter 5 of this thesis, the other contributors to problem complexity beyond the degree of coupling between parts of the system (i.e. aside from how integrated the architecture of the system is) are the novelty of the problem itself and the newness of the technology embedded in the system.

6.2.3 The Influence of Problem Novelty on Knowledge Integration

Similar to the analysis given in the preceding two sections, the problem cases were differentiated by problem novelty, where a high novelty (or new) problem is defined in this thesis as one which has not been encountered before and for which the solution steps are unknown, whereas a low novelty (or old) problem is one which has been previously solved or to which the solution approach is largely known. This typology runs parallel to the routine versus non-routine dimensions used to describe the two main problem solving and knowledge integration modes previously in this chapter, with low novelty problems being addressed through routine problem solving processes, while new problems trigger the use of non-routine strategies, practices and mechanisms for integrating knowledge. Figure 39 below illustrates the differences in the use of knowledge integration channels for both high and low novelty problems.

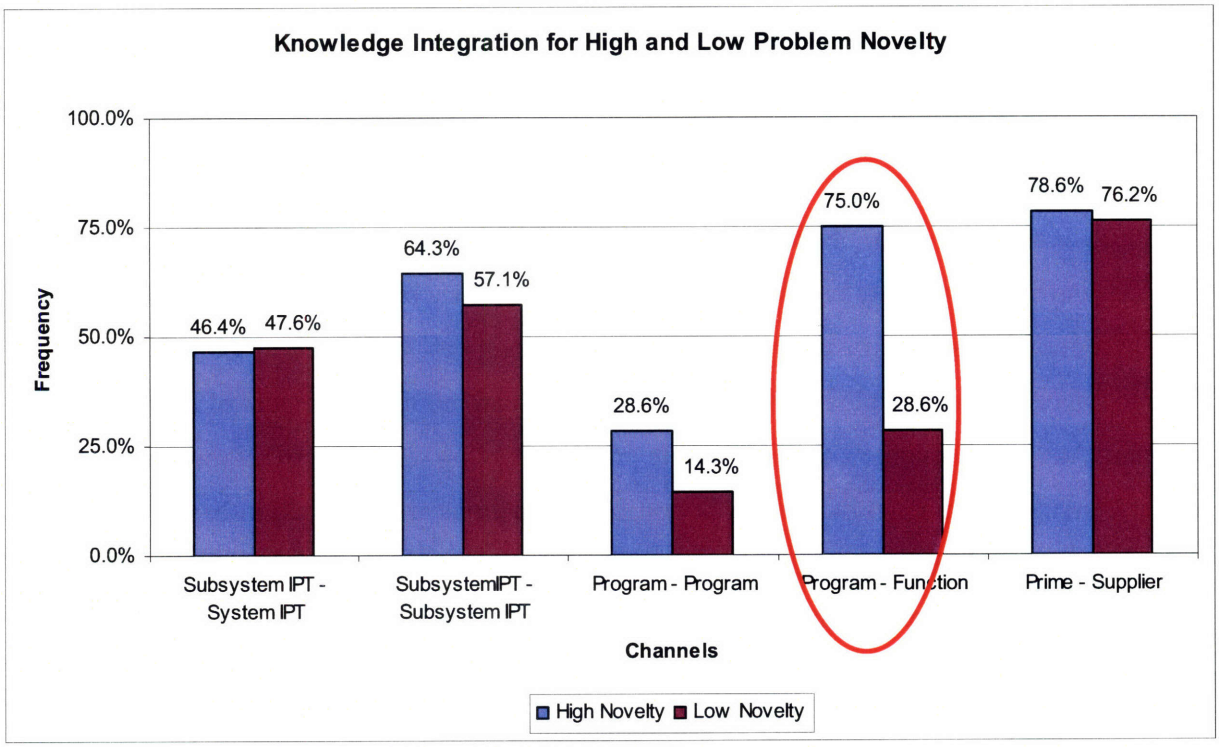


Figure 39: Knowledge Integration by Problem Novelty

The above data shows that overall there is an increase in knowledge interactions across most channels for high versus low novelty problems. This is intuitive since there is a greater need for outside help to solve new problems where the solution is unknown than for problems to which the solution steps are largely known. However the main insight from this comparison is that there is much greater recourse for functional groups in dealing with new problems (87% of all mechanisms are for direct assistance from functions for tackling new problems where the solution is largely unknown) since the functions typically have seasoned problem solvers with long experience and deep expertise with both scientific and engineering knowledge, as already explained in the previous section. This outcome, which is based on the questionnaire data, confirms the previous conclusions reached in the qualitative analysis based on the field interviews conducted separately, specifically that programs reach out to functions for new technical knowledge as well as for help with root-cause analysis and big pictures reviews. However, this makes the relationship with the functional organization a "last recourse" type of interaction, as described by several interviewees: "you go to the functions when you don't know how to solve the problem" and "support from engineering functions is only needed for major problems, this need goes away with experience".

Finally, a perplexing outcome from the data above is the equally high frequency of knowledge interactions for low novelty problems along all other channels besides the program-to-function link. From the questionnaire data and the supporting interviews, the main reason for these high interactions is that in complex development there are is an inordinate amount of information exchange across all channels even in routine problem solving (e.g. 78% of all mechanisms along the prime-supplier channel in low-novelty problem situations are information-based, such as engineering change documents). The types of knowledge and the corresponding mechanisms employed across different channels will be explored in further detail in § 6.2.5

6.2.4 The Influence of Technology Maturity on Knowledge Integration

Another contributor to complexity in problem solving is technology newness, or the degree of maturity of the core technologies embedded in the system. This is because there is a much higher incidence of major problems resulting from untested technologies than from mature

technologies, as already demonstrated in the product development and innovation literature. For the purposes of this research, only the dominant core technology embedded in the system for each problem case is considered, and this pacing technology is classified as either new or old (mature). Figure 40 below illustrates the differences in the use of knowledge integration channels for both new and mature technology problems.

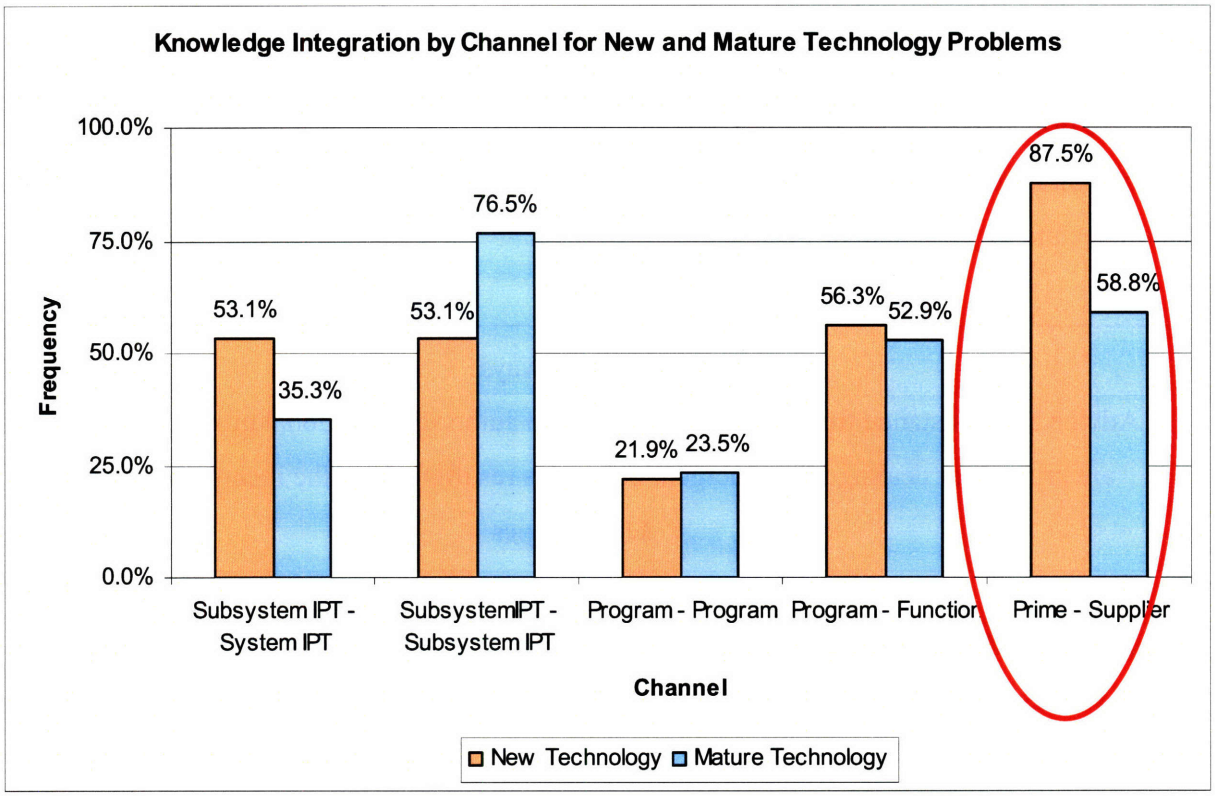


Figure 40: Knowledge Integration by Technology Newness

As already discovered in previous research (Takeishi 2002), knowledge integration between prime and supplier organizations is higher for new technology problems since untested technologies embedded in the supplied subsystems require extensive knowledge interactions to solve emergent problems. Similarly, there is higher system-level IPT involvement to fix new technology problems which tend to have a rippling effect on the rest of the system due to emergent behaviors from unpredictable interactions between the new technology and different parts of the system. In addition, system IPT leads and chief engineers are typically more

involved in solving new technology problems as these problems are considered high-risk and expensive to fix and they usually lead to budget and schedule overruns, thus necessitating higher level involvement. As one respondent from the prime organization put it, "the prime wants to have a say in how the supplier solves the problem since complete solutions are more costly upfront". The frequent use of the subsystem-to-subsystem IPT channel for solving mature technology problems would seem to be counterintuitive at first since mature technologies are well understood, however from the problem solving descriptions in the questionnaire it is clear that this is largely due to extensive exchange of information in support of problem solving (e.g. along the prime-supplier channel, this research found that 75% of all mechanisms are information based, while only 10% are for advice sharing and 15% for direct assistance with mature technology problems).

Key takeaways:

Advice and assistance from higher-level program authority, functional groups and suppliers are critical for dealing with emergent behaviors resulting from the incorporation of new technologies

In conclusion, while the figure above shows a high frequency of interactions for dealing with new as well as mature technology problems across most channels, the richness of interactions for troubleshooting new technology problems is much higher than that for solving problems involving mature technologies.

6.2.5 Dynamic Conceptual Framework for Knowledge Integration

As a prelude to the final refinement of the proposed conceptual framework for knowledge integration, it is necessary to combine the insights about the use of different channels for different problem solving contexts with the data about the use of different strategies, practices and mechanisms for integrating knowledge along those channels. The following Table 37 summarizes the most dominant mechanisms collected in the questionnaire for all 49 problem solving cases, which are listed by highest, second and third highest use per channel.

Table 37: Observed Knowledge Integration Strategies, Practices and Mechanisms

Mechanisms	On Channel #1	On Channel #2	On Channel #3	On Channel #4	On Channel #5
Information Mechanisms					
Requirements, specs, change documents	**Highest**	**Highest**			**Highest**
Problem reports, test data		**3rd Highest**			**3rd Highest**
Advice Mechanisms					
Chief engineer, system IPT lead	**2nd Highest**				
Gurus, graybeards, red teams				**3rd Highest**	
Assistance Mechanisms					
Wizards, tech fellows, SME's	**3rd Highest**		**Highest**	**Highest**	
Co-location, site visits					**2nd Highest**
Troubleshooting meetings, reviews		**2nd Highest**	**3rd Highest**		
Tiger team, taskforce			**2nd Highest**	**2nd Highest**	

Interpreting the data in the table above, it is clear that in non-routine problem solving the knowledge interactions with other programs (along channel #3) and functions (along channel #4) are mostly for direct assistance, namely by calling on experts for their expertise in similar problem solving events, and by forming multi-disciplinary special action teams. In contrast, knowledge interactions across all other channels are dominated by information flow, due to the iterative nature of non-routine problem solving (e.g. during root-cause analysis, a host of problem reports and test data are generated and exchanged between all parties involved in problem solving); however, the frequency of use of advice and assistance mechanisms was close behind the integration of information, with mediation by IPT leads, system architects and chief engineers being the primary strategy employed along the subsystem-to-system IPT channel (channel #1), while troubleshooting meetings were most frequently used to diagnose and solve problems with other subsystem IPT's. Last but not least, the prime-supplier channel was used most frequently to move the locus of problem solving to the source of the problem, or

alternatively to where the knowledge required for the solution was most concentrated, depending on the problem. These conclusions are illustrated in Figure 41 below:

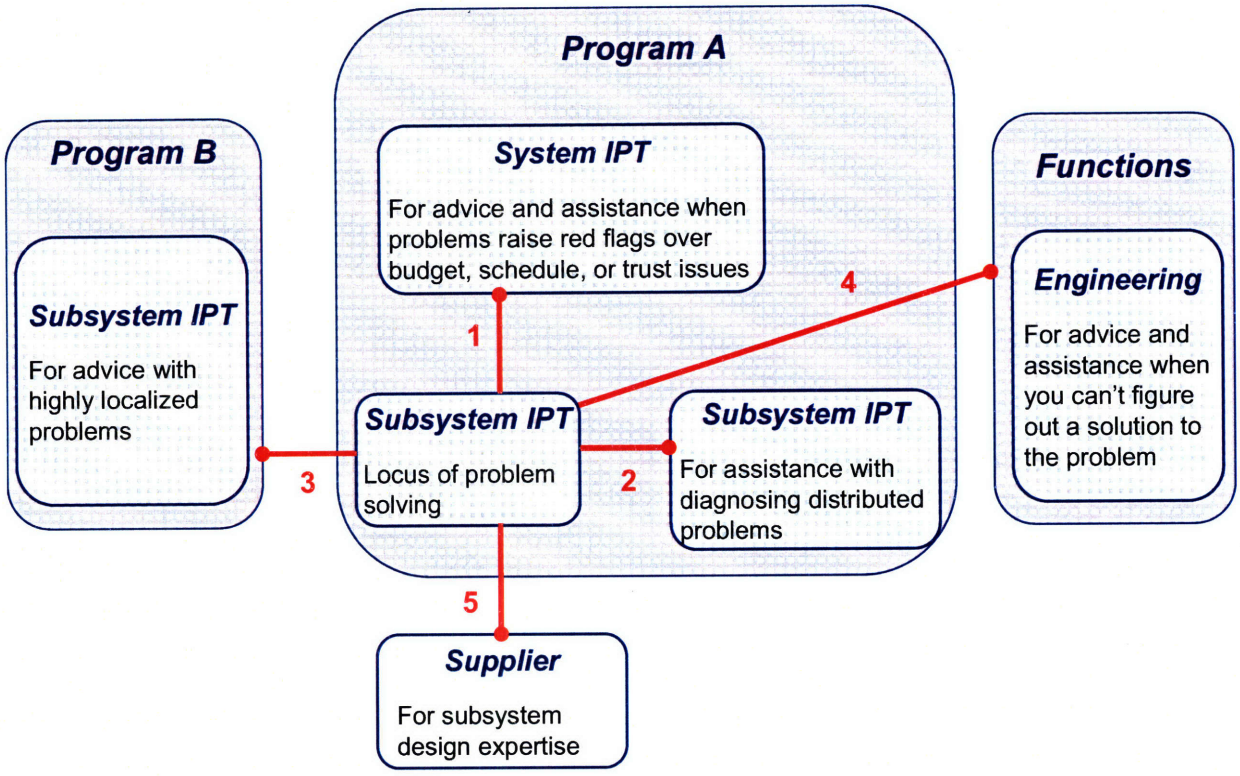


Figure 41: Dynamic Framework for Knowledge Integration in Complex Problem Solving

Combining the insights from the previous steps of the quantitative analysis with the data about the most frequently used strategies and mechanisms, a final refinement of the knowledge integration framework is proposed and shown in Figure 42 below:

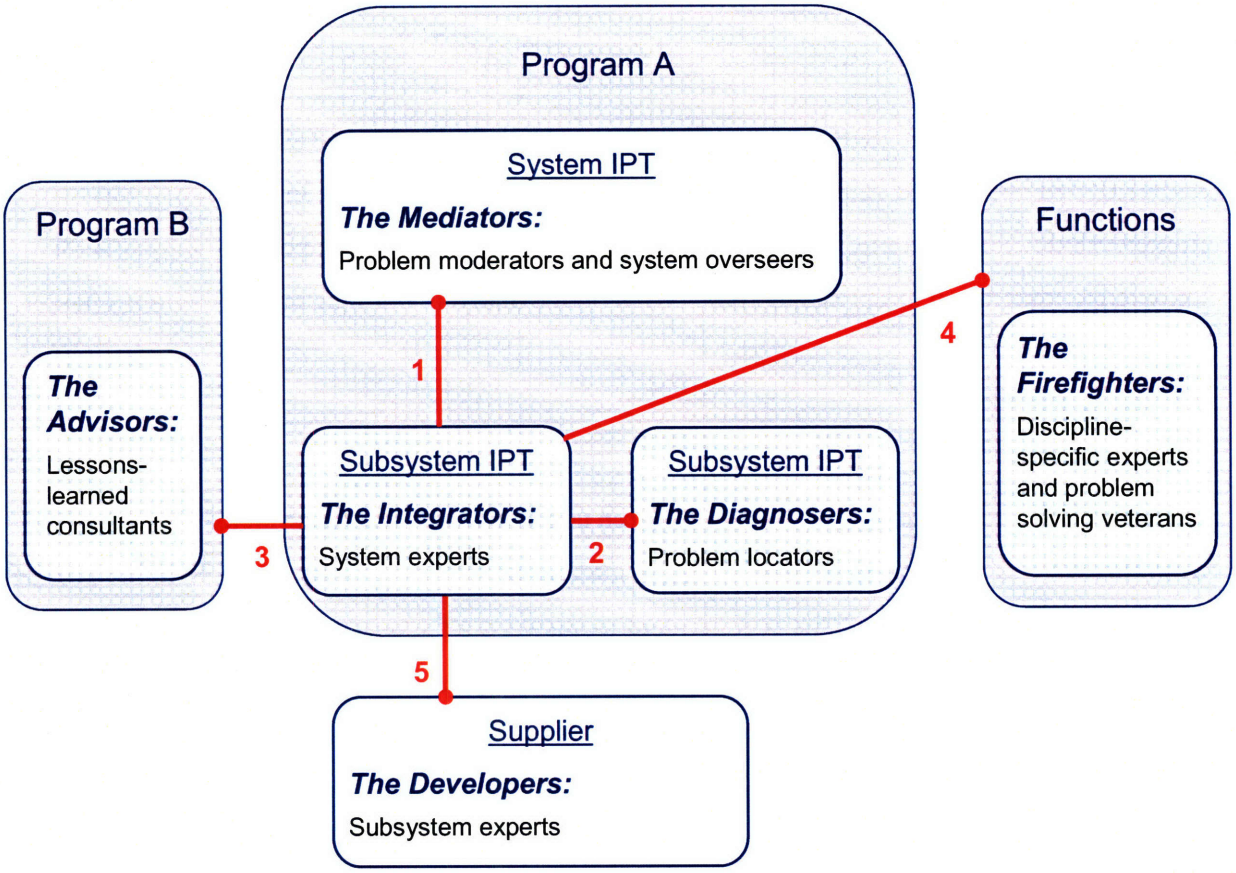


Figure 42: Final Framework for Knowledge Integration in Large Scale Complex Problem Solving Environments

7. CONCLUSIONS

This chapter presents the main findings from the research on the process of integrating knowledge in large-scale complex product development environments. In addition to the framework developed in this thesis and presented in Chapter 6, some important heuristics for knowledge integration are presented here based on the analysis findings in the previous chapter.

7.1 Major Findings

In line with the initial motivation of this research as posited in § 1.2, the principal contribution from this work is in describing the knowledge integration process in large scale complex product development environments. This is equivalent to answering the question: how is knowledge integrated under different problem solving conditions? The answer to this question, which was illustrated in the framework for knowledge integration developed in Chapter 6 of this thesis, can be summarized as follows: first, in routine problem solving situations where the solution approach to the problem at hand is largely known, this research has shown that knowledge integration can be efficiently and effectively accomplished through mostly explicit and formal integrative mechanisms, especially through sharing lessons learned which were found to be critical for dealing with complexity in this context. In such routine conditions, the approach to solving the problem follows systematic and programmatic steps where the lines of communication are tight, using mechanisms such as formal share sessions, design reviews and information systems.

Second, in non-routine problem solving situations where the "aha" state of problem solving is not obviously discovered, problem solving is highly iterative until the root cause of the problem is identified (which is typically accomplished by duplicating the observed failure through numerous trial-and-error attempts), and the main finding from this research is that in addition to explicit knowledge, the integration of tacit know-how and expertise is critical for dealing with the emergent behaviors commonly encountered in this context. In these non-routine conditions, the approach to solving the problem involves a lot of speculation in trying to identify the problem root cause, which requires broad lines of communication to exchange informal advice

and suggestions. In many cases, this leads to direct assistance from outside experts in the troubleshooting process, therefore increasing the exchange and application of tacit knowledge during problem solving. That is not to say however that explicit knowledge is not important in non-routine situations, indeed it was found in this research that testing is extensively employed to reduce speculation with real data, thus leading to the frequent exchange of explicit information along internal and external channels. By the same token, tacit knowledge is also essential in routine problem solving and some of the most common mechanisms discovered in this research are communities of practice, boundary spanners and frequent design reviews.

However, a main conclusion from this research remains that different problem solving conditions require the use of different channels, strategies, practices and mechanisms for knowledge integration, with some of these being dominant in particular situations. Furthermore, this research has shown that different characteristics of the product under development force the prioritizing and/or use of particular knowledge integration channels and mechanisms. These findings constitute an original contribution to existing theory in as much as they demonstrate how the engineering artifact drives the organizational system through the knowledge integration process. Along those lines, and building on the main findings from the analysis in the previous chapter, the following heuristics for knowledge integration are proposed in Table 38 below:

Table 38: Heuristics for Knowledge Integration by Product Architecture and Technology Newness

	Modular Architecture	**Integral Architecture**
New Technology	**Problem solving with suppliers is critical** Direct involvement of functions and other programs is very useful (Advice, assistance on ch. #4, 5)	**Problem solving with other subsystem teams is critical** Direct involvement of functions and suppliers is very useful (Assistance on ch. #4, 5)
Old Technology	**Testing with suppliers is sufficient** (Information on ch. #5)	**Testing with other subsystem teams is sufficient** (Information on ch. #2)

As these general heuristics show, the integration of knowledge for problems involving mature technologies is generally limited to the exchange of information along a few channels, while new technology problems typically require extensive knowledge interactions across multiple channels using informal advice and direct assistance in the process. However, the direction and extent of knowledge integration in either technology regime also depends on the type of problem being tackled (e.g. whether it is a subsystem design problem requiring design knowledge from the supplier base, or a system integration problem requiring architectural knowledge from sources in the prime organization), and also on the novelty of the problem itself. Heuristics for problem solving involving different types of problems with different levels of problem novelty are proposed in Table 39 below:

Table 39: Heuristics for Problem Solving by Problem Type and Novelty

High Problem Novelty	**Localized or distributed across many subsystems** Typical fix: requirements changes, hardware or software redesign: need direct assistance from suppliers, informal advice and direct assistance from other programs and functions	**Distributed across many subsystems** Typical fix: requirements and specs changes, hardware and software redesign: need information sharing and formal advice from suppliers, direct assistance from functions, concurrent inter-IPT problem solving
Low Problem Novelty	**Localized in one subsystem** Typical fix: requirements changes and hardware redesign: need intra-IPT problem solving	**Distributed across few subsystems** Typical fix: interface specifications changes and software redesign: need inter-IPT information transfer
	Design Problems	**Integration Problems**

Using a simple metric to measure problem complexity as a combined factor of problem novelty, problem spread[48] and technology newness (where a highly complex problem is equivalent to a new problem affecting multiple subsystems and involving untested technologies, whereas low

[48] Problem spread is a measure for whether the problem is localized or distributed across several subsystems, which is an indication of total system complexity since a high number of affected subsystems indicates a high degree of coupling and therefore a high level of total system complexity

problem complexity is equivalent to a localized problem to which the solution steps are largely known and which involves mature technologies), the following heuristics can be proposed in Table 40 below:

Table 40: Heuristics for Knowledge Integration and Problem Solving by System Architecture and Problem Complexity

	Modular Architecture	**Integral Architecture**
High Problem Complexity	**Hard to solve problems (innovative solutions needed)** • "Rich" mechanisms with suppliers are critical • Need formalized processes for advice sharing with other programs • Need process checks for triggering help/mediation	**Hard to diagnose and solve problems (iterative trial-and-error needed, solutions at multiple fix points)** • Rapid mobilization capabilities for assistance are critical • Need dedicated functional resources to reduce reliance on suppliers
Low Problem Complexity	**Easily solved problems** Quickly solved problems are quickly forgotten - need to capture knowledge from "easy solutions" in information systems	**Easily diagnosed problems** Over-reliance on IT systems leads to "false-positives" - need to supplement with tacit knowledge interactions between subsystem IPT's

The most notable rules of thumb from the table above are for the high problem complexity cases shown in Table 40 above, where process checks for triggering help with localized problems early-on are the key to avoiding problem solving in isolation, which is typical behavior by the naturally self-sufficient IPT's, and which leads to point fixes instead of systemic solutions. Similarly, having dedicated functional resources is critical for effective assistance with tough distributed problems involving multiple subsystems, especially that these problems take extensive iterations to diagnose and locate their root-cause, which typically leaves little time and resources for developing long-term solutions rather than cheaper short-term fixes.

To further expand on the main conclusions and heuristics above, and since a central aspect of the knowledge integration process is the gathering of knowledge distributed across organizational boundaries (as per the original definition for knowledge integration proposed in § 2.1.2), it is appropriate to interpret the main new findings from this research in terms of the *permeability* (or lack thereof) of some of those boundaries as conceptualized in the knowledge integration framework developed in this research. The expanded conclusions are discussed in detail in the following sections.

7.1.1 Impermeable Cross-Program Boundaries

To start with, and based on the analysis in the previous chapter, it is amply evident that a major impediment against the efficient and effective integration of knowledge in large-scale organizations lies in the impermeability of cross-program boundaries due to a host of barriers such as proprietary and classified restrictions (see Table 47 in Appendix B for a list of program-to-program barriers), but especially due to a mentality of local program optimization as opposed to collaborative arrangements leading to global benefits for the entire organization, much less for the overall organizational network. This finding is in line with insights from the knowledge transfer literature about the "stickiness" of knowledge internally to the organization as opposed to knowledge "leakiness" across external boundaries (Von Hippel 1994), however the findings from the current research add a new layer of specificity to existing insights by singling out the cross-program interface as the 'weakest link' for knowledge integration, and more specifically for the integration of problem solving know-how, skills and expertise across programs. And while a few enablers (indeed, very few) were identified that counter the inward focus of programs on their own goals (see Table 46 in Appendix B for a list of program-to-program enablers), such as incentivizing program managers to focus on corporate performance beyond their own program, the reality remains that the strategies employed to promote direct knowledge flow across programs lack the 'teeth' (in terms of appropriate mechanisms) for achieving this goal in practice. For example, it was found that cross-program knowledge integration through information systems is insufficient and inadequate in major problem solving situations, with costly state-of-the-art systems such as multi-program lessons learned databases being rarely used in non-routine problem solving due to the difficulty of mining their contents, in addition to issues

of information obsolescence and lack of verification of these contents. They are therefore rarely useful in practice and costly to setup and maintain. Similarly, while process standardization is in principle a sound strategy for integrating process knowledge between programs, it was found that 'blind' standardization (e.g. the propagation of standards without the accompanying lessons learned during the development of those standards) was a 'knot' for problem solving since it can propagate potential problems across multiple programs. It can thus be concluded that explicit-based mechanisms alone, such as information systems and standards, are not sufficient for making impermeable boundaries more porous, and it follows that these must be supplemented with mechanisms for integrating tacit learning and know-how between individuals.

But when it comes to the integration of such tacit knowledge across those same boundaries, a main finding from this research is that the channels and mechanisms used for that purpose are mostly indirect and mediated by the functional organization, such as with functional groups moving experts and personnel to where they identify a greater need for them, which is inefficient and in some cases ineffective relative to the use of direct channels between programs (Cohen and Levinthal 1990), as already discussed in § 2.1.9. This was confirmed by field observations where several interviewees pointed to numerous shortcomings from having the functional groups decide which personnel to move and when to move them out of the program. It was also observed however that there are some effective strategies commonly employed by the functional organization which can compensate for these shortcomings, such as in conducting formal design reviews at critical milestones, or in overseeing non-advocate reviews by one program of another where tacit knowledge is efficiently and systematically transferred directly between programs. Functions are also neutral mediators in the formation of special problem solving teams from a pool of program and functions experts for dealing with major troubleshooting events. These strategies allow programs to overcome some of the barriers mentioned above which prevent the integration of learning and expertise between them; however, a common view persists that these strategies do not constitute 'true integration' compared with direct relationships (i.e. the outcome is suboptimal compared with direct knowledge integration), and that going through the functions is costly to the organization as a whole in terms of the financial and human resources expended for that purpose, and to the programs in terms of the time and effort spent in the process.

7.1.2 Indirect Relationships between Programs and Functions

A related finding on knowledge integration across internal firm boundaries is the 'last recourse' relationship with functional groups when it comes to asking them for problem solving support in major problem situations. This was attributed to 'engineering pride' in problem-solving and the fact that engineers want to tackle challenging problems themselves rather than asking for help when they encounter difficulties. However, a recurring symptom resulting from this cultural barrier is that in major problem solving situations, soliciting help from the functions typically comes too late in the process such that impacts to schedule and budget are unavoidable at that point. It follows that the subsequent integration of knowledge for problem solving under rushed conditions is itself suboptimal, both in terms of the outcome (e.g. limited learning) and the efficiency of the process itself (such as being forced to use less efficient mechanisms in rushed circumstances). It is tempting here to blame the entire firefighting phenomenon on the engineering culture (which was sometimes observed in this research), however according to previous research it is instead the susceptibility for poor resource planning in large scale complex product development environments, especially in terms of thinly spread problem solving resources (typically distributed across parallel projects) and the panicked reallocation of those resources in major problem situations, which are the main contributors to continuous firefighting in this context (Repenning 2001). This is why a commonly expressed desire among interviewees was for better resource planning and process metrics from the functional organization in order to "prevent the cart from going into the ditch in the first place, not just get it out", and a common realization was that "identifying symptoms (before the problem occurs) is more important than using lessons learned (after the fact)", especially in parallel development and concurrent engineering contexts which are most prone to firefighting. Along those lines, this research identified some positive policies by the functional organization addressing the firefighting phenomenon, for example in encouraging and developing those innovations which reduce the likelihood of problems occurring down the road, and rewarding individuals who develop workarounds that help to extinguish existing 'fires'. It was found however that while both of these policies are helpful in dealing with firefighting, the reactive policy of rewarding firefighters has the potential of encouraging counterproductive behavior by individuals seeking reward and recognition who proceed to create their own fires in order to extinguish them later. This leads to

a self-reinforcing feedback loop of arson-and-firefighting which defeats the original purpose of extinguishing existing fires. I thereby conclude that the better policy is the proactive approach which ultimately leads to more robust and controlled problem solving processes overall.

7.1.3 Arm's Length Relationships between Prime and Supplier

In terms of the permeability of external boundaries, it is expected that the "leakiness" of knowledge across prime-supplier boundaries would naturally promote its flow between separate organizations. However, it has been shown in previous research that knowledge integration across organizational boundaries is susceptible to issues of transparency (or the willingness to share knowledge) and receptivity (or the ability to absorb knowledge) (Larsson, Bengtsson et al. 1998), and that inter-organizational learning faces fundamental dilemmas such as the need for each firm to protect its own core competency, and the tendency of some firms to engage in opportunistic learning (Dyer and Nobeoka 2000). These and other barriers discovered in this research (see Table 47 in Appendix B for a list of prime-supplier barriers) hinder the effective integration of knowledge across external boundaries. In particular, this research has shown that prime-supplier boundaries are not effectively negotiated in large-scale networks, due in large part to the higher number and severity of the barriers encountered at this scale, such as for example the increasing likelihood for restrictive government regulations, thus necessitating more 'workarounds' and mitigation strategies to counter them (see § 7.3 for recommendations).

For example, in the case of military avionics, the high complexity of the systems under development leads to a growing need for large-scale networks in order to accomplish the development task, which in turn increases the likelihood of involving foreign suppliers in the development process. One such example is the ITAR regulatory restriction on the export of weapons technology to foreign entities which severely restricts the integration of knowledge with foreign suppliers. In addition, the influence wielded by the bigger firms in large-scale networks makes for uneven relationships with smaller suppliers, forcing the latter to be more protective of their interests and less trusting of the other members of the network. This confirms earlier insights that the effective integration of knowledge in large-scale networks is contingent on relationships of trust between different organizations and organizational clusters (see § 2.2.4).

As stated by one of the interviewees, the lack of trust "typically leads to resistance against the use of effective (knowledge integration) mechanisms"

A clear manifestation in this research of the arm's length relationship between prime and supplier is in the dominance of explicit knowledge integration along this channel, especially involving constantly changing requirements and specifications (as shown in Table 39), which is due to poor supplier integration in the development process, particularly at the early stages of the requirements definition phase. This was confirmed by respondents who stressed "the need for close supplier membership to arrive at well-defined, concise and complete requirements". However, not all explicit knowledge integration along this channel is about the exchange of requirements-related information, and it was found in this research that much of the cross-program integration of design knowledge and new technological knowledge happens through common suppliers as the intermediaries. It is therefore apparent that routine knowledge integration across programs is more efficient and effective at the supplier level than at the prime organization level, meaning it is less costly than knowledge integration through the functional groups and more beneficial for the programs in terms of acquiring new design knowledge than mere process standardization emphasized by the functions. This however does not come without risks to the prime organization of becoming 'hostage' to key suppliers responsible for common subsystem modules across multiple programs, and it is evident from the analysis of the data in this research that prime organizations are already highly dependent on their suppliers for most new design and technological knowledge during problem solving. Along the same lines, this research has shown that major suppliers are increasingly acquiring architectural knowledge from the prime during problem solving, giving them the potential to develop their own competency in system integration, and to threaten the prime's competitive position (Henderson and Clark 1990).

In summary, this section has summarized the new contributions from this research in terms of major insights about the knowledge integration process in large-scale complex product development. The conclusions presented here are in line with the initial research motivation of providing in-depth insights about the process of integrating knowledge in large-scale complex problem solving environments. Some of the main implications for problem solving and organizational integration discovered in this research are also presented in the following sections.

7.2 Research Implications

As already demonstrated in the data analysis chapter of this thesis, the processes of knowledge integration and problem solving are closely intertwined, with both influencing and being affected by the organizational context. In the course of this research, several macro-level implications emerged that inform the problem solving process in complex product development and the structuring of the organizational environment, as discussed below.

7.2.1 Implications for Complex Problem Solving

It is well established in the literature as reviewed in § 2.2 of this thesis that different aspects of complexity, whether related to the actual problem at hand, the engineering artifact or the organizational context, drive the problem solving process; however, there are very few insights about the actual implications of complexity on problem solving in practice.

For example, examining the heuristics in Table 40 above, it is clear that for highly complex problems, there is a need for new tacit knowledge and expertise from multiple sources inside and outside the developing program, such as experts and special action teams, which translates to a need for 'mobilizing the troops' to help with problem solving as early in the process as possible; however when dealing with integrated architectures in particular, problem diagnosis becomes a highly iterative process involving repeated trial and error attempts at locating the root cause of the problem due to coupling between different parts of the system, which increases the time and effort needed to address the problem at hand and thus compromises problem solving efficiency and effectiveness. This is compounded in large-scale organizational environments where typically large geographic distances separate the developing teams. As one interviewee put it "you can't solve any problem without flying everywhere to talk to everyone because of tight integration". Therefore, a major implication for complex problem solving from this research is that integrated architectures make the developing teams lose rapid troubleshooting capability, thus increasing the likelihood of firefighting which is already prevalent in complex product development environments, and as a result necessitating the use of more costly mechanisms for

integrating knowledge such as frequent site visits, or even co-location of geographically separated teams.

A related implication for complex problem solving involving highly integrated systems is in the conclusion that some of the knowledge integration mechanisms employed in this context are conducive to counterproductive outcomes, especially those which are designed to integrate explicit knowledge across boundaries. For example, a commonly used mechanism for integrating knowledge about problems across team boundaries is a common problem reporting system allowing engineers to report anomalies observed during the system integration phase, which become visible to other teams responsible for interrelated parts of the product under development. But since problem solving is highly iterative in integrated systems, this type of mechanism can lead to 'finger-pointing' and 'throwing-over-the-wall' behaviors between teams involved in locating the root cause of the problem, as these impersonal systems lack the richness of tacit knowledge mechanisms such as face-to-face meetings which encourage dialogue and collaborative conflict resolution. In addition, the lack of oversight and the inability to have 'live' process checks over the use of such impersonal systems creates the potential for reporting 'false positives' as engineers mistakenly report what they think is an anomaly but which in reality is not. This in turn translates to injecting unnecessary requirements changes into the development process, thereby leading to a wasteful self-reinforcing behavior known as "requirements creep". As a result, the need for augmenting explicit-based mechanisms with richer individual interactions becomes increasingly critical when dealing with highly integrated architectures.

Conversely, when dealing with complex problems in modular architectures, it is well established that the process of problem solving is typically easier than for problems encountered in integrated architectures since the former are typically localized in one module or a few interdependent modules, which makes these problems easier to diagnose (note however that this does not mean that problems in modular architectures are less complex or less difficult to find a solution for than problems in integral architectures, just that the problem solving process is typically less iterative since the problem root cause is easier to find). However, it was discovered in this research that problem solving in modular architectures requires extensive knowledge integration across external boundaries, namely along the prime-supplier boundary

and across program boundaries, both of which are difficult to navigate as already explained in the previous section. This makes localized problems especially hard to solve, which is contrary to accepted wisdom as detailed in the argument above. However, this finding supports conclusions from previous research that teams specialized in modular systems development are less effective at interacting across organizational boundaries than teams experienced with integrated systems (Sosa, Eppinger et al. 2004).

Another implication from this research related to complex problem solving is in the finding that quickly solved problems are often quickly forgotten, which means that knowledge is captured only after 'painful' problem solving events (i.e. where significant time and effort are expended in problem solving) but not for solutions which are less painfully developed. However, the value of captured knowledge does not necessarily correlate with the magnitude of the problem solving event, and it can indeed be the case that solutions which were easily developed in some circumstances may not be as easy to rediscover from scratch in a different context or by different individuals, which means it is just as important to capture knowledge about easily-developed solutions as about 'big' problem solutions. In fact, the lack of knowledge capture for minor problem solving events may help to explain a recurring complaint observed in this research about the inadequacy of information systems, such as shared databases, in terms of their usefulness for complex problem solving due to usually poor knowledge content as reported by interviewees. However, when considering the learning-and-forgetting symptom for what are mistakenly considered as 'minor' or 'small' problems, it becomes apparent that information systems are sometimes blamed for failures elsewhere, namely in the lack of capturing solutions after short problem solving events. In other words, this finding makes it equally plausible to believe that when a knowledge search tool fails to return any useful results, it does not necessarily mean that this is due to shortcomings in the tool's capability at properly mining the data, but that it could simply be due to the lack of available information caused by failures in knowledge capture elsewhere.

Finally and along similar lines, it was frequently repeated in the interviews that problem solving with suppliers is "highly collaborative and informal at the early stages of problem diagnosis, but becomes more formalized as you get closer to the solution". The implication from this is that

once the problem root cause is identified, the most valuable outcome of the knowledge integration process in terms of knowledge about the solution is held closely by both parties, and that the visible collaboration on the surface can be a deceptive indication of open knowledge sharing. While this may be a good strategy for each party to protect its core knowledge and still be able to collaborate with one another in problem solving, it is nonetheless a short-sighted strategy at the level of the organizational network, since it compromises network learning and encourages opportunistic behavior by individual network members. A more productive strategy in this case would be for the focal firm (or prime organization) to act as a knowledge sharing catalyst and a central node for knowledge transfer in order to increase the knowledge of the overall network instead of individual network members. This aspect of knowledge integration will be explored further in the following section on organizational integration.

7.2.2 Implications for Organizational Integration

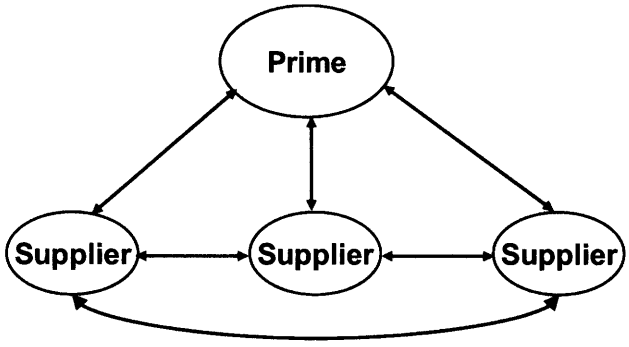
The discussion in § 7.1.1 focused on the permeability of organizational boundaries and those strategies and mechanisms discovered in this research that promote knowledge flow across internal and external boundaries. However, it is important to recall here that the objective of knowledge integration is not to increase knowledge flow at any cost, such as in a "zero-sum game" where the gain in knowledge on one side of the boundary is a net loss on the other side, and a main insight from this research is that the overall effectiveness of the knowledge integration process is significantly dependent on the choice of appropriate mechanisms for the problem solving context at hand. As such, there are no "one-size-fits-all" types of mechanisms for integrating knowledge, and any mechanism can become counterproductive in the wrong context. This was evident in the interviews where some of the most commonly cited mechanisms (i.e. the 'best in class' in use at the organizations surveyed) for integrating knowledge under routine problem solving conditions, namely job rotation and moving experts around, were also said to be the means for "knowledge stealing" and "cherry picking" by one program from another, specifically in instances when the contributing program could not afford to lose its experienced talent, such as in non-routine firefighting situations or at critical phases of product development. This means that mechanisms which are appropriate for integrating knowledge in routine problem solving conditions may not be appropriate in non-routine

conditions. In addition, and from the larger organizational perspective, only those mechanisms that promote mutually beneficial knowledge interactions for the individual parts can lead to an effective knowledge integration outcome for the whole. Yet as pointed to above, there is no 'magic bullet' when it comes to the choice of specific means for knowledge integration; instead these choices should be based on a portfolio approach, where a series of complementary strategies and mechanisms are made available by the organization in order to deal with different problem solving contexts. From the perspective of organizational integration, previous research has shown that bridging differences between interdependent parts of an organization requires a combination of formal and informal structures using both tacit and explicit knowledge integration mechanisms (Daft and Lengel 1986). Therefore it can be concluded that in large-scale complex product development environments where integration of separate but interdependent entities is critical for efficiency and effectiveness in problem solving, organizations are best served to have dynamic portfolios of integrative mechanisms which can be adapted to different problem solving contexts. At the macro (strategic) level, this gives the organization 'strategic flexibility' in terms of allowing it to face uncertainty from external markets more effectively (Sanchez 1997). At the micro (operational) level, this affords the organization the ability to bridge discontinuities internally. For example, to compensate for the knowledge drain from older programs, an organization with a diverse portfolio of integrative mechanisms can enable advice sharing between old and new programs through both formal and informal mechanisms such as explicit-based expertise locator systems and tacit-based social networking mechanisms. This helps to control the need of new programs for senior experts and thus limits the severity of the exodus of expertise from older programs.

In parallel to implications relating to complex problem solving in integrated architecture regimes as discussed in the previous section, this research has found that when dealing with modular architectures, there is a tendency for subsystem teams to work in isolation from each other due to the self-sufficient nature of the IPT structure, therefore leading to multiple point solutions instead of a single complete system-level fix. This means that the solutions developed are often 'satisficing' instead of being optimal, and more so when recluse IPT's are late in asking for outside advice and assistance, thereby limiting the time available to external experts for tackling the problem at its root and developing a complete solution. This in turn feeds the firefighting

phenomenon as more resources have to be diverted to deal with the problem at the last minute than would have been needed at an earlier stage of problem solving. The implication of this for organizational integration is that project-based organizational structures, which are commonly employed in complex product development and where autonomous IPT's are at the center of problem solving, have the potential of reinforcing the isolation of developing teams to the point of losing focus on system-level issues, thus making it necessary to have redundant and expensive boundary spanning mechanisms such as taskforces, liaison individuals and full-time integrators.

Finally, as discussed in the previous section, just like individual organizations have to continuously develop and protect their core competency in order to sustain competitive advantage, so do organizational networks, with the difference that the challenge in the latter is a bigger one of evolving collective learning capabilities through mutually beneficial relationships. In previous research, the relationships between firms have been explored, including the vertical relationships between the focal firm and its suppliers and the horizontal relationships between suppliers in the same network (Dyer and Nobeoka 2000; Takeishi 2002), and it was found that networks which are effective at knowledge sharing are those with strong multi-lateral ties for transferring both tacit and explicit knowledge, as illustrated in Figure 43 below:



Adapted from Dyer and Nobeoka, 2000

Figure 43: Strong Ties in Knowledge Sharing Networks

However, organizational networks have not been further dissected down to the program level to recognize the discontinuities in smaller clusters in the network (where a cluster is made up of a major program and its suppliers), as shown in Figure 44 below:

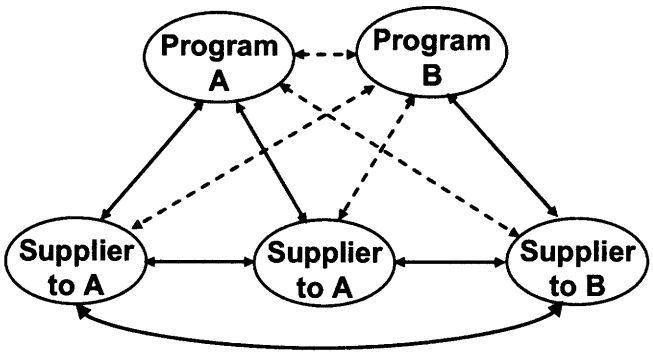


Figure 44: Weak Ties (Dotted Lines) in Knowledge Sharing Networks

As already discussed in previous sections, this research has shown that horizontal integration channels across programs within the prime organization are walled off by a host of barriers and the integrative mechanisms used along those channels were found to be limited and generally inadequate for supporting complex problem solving efforts. In addition, collaboration along prime-supplier boundaries is for the most part not conducive to network learning, especially in terms of supporting commonality across programs (recall from § 6.1.2 that indirect relationships between programs mediated by functional groups were found to be inadequate for the flow of tacit knowledge and a hindrance to cross-program commonality). Therefore, while on the surface the organizational network appears to have strong ties with few structural holes, the reality at the level of network clusters is different, with multi-level ties going across clusters (illustrated with dotted lines in Figure 44 above) being the weaker links. It follows that in order to effectively foster network learning it is necessary to further expand the scope of knowledge integration to include inter-cluster ties, which requires a paradigm shift from protecting individual core competencies for each cluster to collectively evolving *network-level competencies* for the whole. For example, it was discussed in the previous section that prime and supplier organizations protect their individual core competencies by formalizing their collaboration during problem solving the closer they get to the solution (e.g. through contractual agreements). A paradigm shift would require closer integration throughout the problem solving process and a formalization of the outcome instead of the process itself, so that knowledge about the developed solutions can ultimately be disseminated to other network members. Here the

focal firm is best positioned to act as a catalyst for combining expertise from multiple members and sharing them across the network, while at the same time fostering trust and shared commitment and controlling for opportunistic learning by individual members.

7.3 General Recommendations

This section provides a summary of recommendations distilled from the analysis and the previous discussion which are generalizable across organizational and industry contexts, as follows:

- Break down the walls: as already discussed in detail in the previous sections, a major barrier against efficient and effective knowledge integration for complex problem solving is the impermeable program-to-program interface. It is therefore necessary for large-scale organizations involved in complex product development to break down their program silos through a rich portfolio of complementary mechanisms for integrating both tacit and explicit knowledge, in order to always have the necessary mechanisms for different problem solving conditions (e.g. databases are inadequate for integrating tacit design knowledge across programs, need tacit knowledge mechanisms such as formal access to other program experts)

- Extend prime-supplier collaboration: this research found that prime-supplier collaboration in problem solving is limited to the early stages of problem solving, as discussed above. However, as widely established in existing theory and as widely observed in practice, protectionist policies against knowledge leakiness across external boundaries are ineffective and counterproductive as they hinder organizational and inter-organizational learning. It is therefore necessary for the focal firm (or prime organization) to act as a knowledge sharing catalyst and a central node for knowledge transfer in order to increase the knowledge of the overall network instead of individual network members.

- Share lessons learned formally and informally: common wisdom dictates that the more you ask the more you shall receive (and the less you have to keep wondering), therefore sharing lessons learned and giving advice are critical for problem solving, and it has been shown in

this research that sharing lessons form previous major troubleshooting events is a critical strategy for dealing with complexity. Multiple strategies for sharing lessons learned should be devised for supporting both routine and non-routine problem solving, with the caveat that while the former can be easily and effectively supported through efficient devices ranging from informal 'brown bag' lunches to formal dedicated repositories, the latter (requiring mostly advice sharing and assistance) is more difficult to address. In fact, non-routine problem solving is often 'left to its own devices', and some of the afforded mechanisms, such as advice sharing, can even carry negative connotations[49]. However, when considering that advice sharing can often be a prelude to further engagement in problem solving, it becomes clear that this powerful mechanism is not only useful for integrating valuable experience but also for establishing interest and trust between parties, both of which are essential for collaborative problem solving. Therefore, enabling formal and informal advice sharing is critical for sharing lessons learned and dealing with complexity in problem solving.

- Establish formal knowledge search capabilities: it is said that it's not (only) what you know that counts, it's (also) who you know, and while having the right mechanisms for integrating expert knowledge is critical in complex problem solving, knowing who to ask is even more important. Knowledge search mechanisms are thus a necessary prerequisite for effective knowledge integration, and it was found in this research that a major barrier against advice sharing is being able to find out who the right expert is in the first place. This requires more than just informal social networking mechanisms as was commonly observed in this research, but also formal search mechanisms provided by the organization, such as expertise locator systems, since most junior engineers who require the most advice lack the rich social networks needed to get to the right senior experts.

- Maintain a common knowledge base: yesterday's knowledge doesn't always solve today's problems, and making program boundaries more porous for the flow of knowledge is not useful without having parallel strategies to ensure a common knowledge base; in other words, it is not useful to have access to irrelevant or obsolete knowledge in other programs,

[49] For example, advice sharing in certain contexts can be considered as a diluted form of knowledge integration where the advice giving party does not want to provide much help or get deeply involved in problem solving.

this is why it is important to have parallel strategies for knowledge commonality across programs as a precondition for meaningful integration, such as the prototyping of future designs by older programs for the benefit of newer programs, or the building-in of design flexibility (e.g. upgradeability and backward compatibility) to ensure knowledge adaptability for different problem solving contexts.

- Break the vicious circle of ignorance: there is no bliss in being a novice in the midst of a complex engineering environment, and one of the fundamental failures of the job rotation mechanism as commonly implemented in large-scale organizations is that it does not afford novice engineers the opportunity to tackle the newest/hardest problems since the objective from this mechanism is to provide breadth over depth of knowledge. As a result, junior engineers fail to acquire a deep expertise with particular types of problems, especially that a typical job rotation lasts for a short 2-3 year tenure. This means that the engineer's eventual repertoire of personal lessons learned will be limited and not reflective of the number of years of work experience they will have accumulated. It also means that their general problem solving skills (regardless of the field of expertise) will be relatively underdeveloped. It is thus necessary to compensate for these shortcomings by 1) allowing junior engineers to participate in special action teams for major problem solving events alongside seasoned experts in order to develop their general troubleshooting skills, and 2) exposing junior engineers to different subspecialties within the same field (e.g. different subspecialties in avionics such as microwave signals, detection, infrared) instead of only across different disciplines (e.g. avionics, structures, propulsion) in order to deepen their specific expertise. This can be efficiently accomplished through job rotations across different subsystem teams for the same system instead of different teams/divisions for distinct systems. These suggested enhancements to the job rotation mechanism can provide junior engineers with deeper expertise in particular fields as well as better problem solving skills in general.

- Fix problems at the foundation: with the tendency of IPT's in complex product development to focus on point problems instead of system-level issues, there is a potential for problems to become recurring or embedded in the system, thereby requiring a fix at the literal 'root' cause of the problem, which is typically the structural foundation of the IPT organization itself.

One obvious and traditional option is to entirely move away from IPT's to functionally organized teams, which carries with it its own set of disadvantages as discussed in § 2.3.4. This option was frequently cited as the most desirable by interviewees as it affords the integration of similar expertise across programs and helps in avoiding isolated problem solving. However, from the analysis of the data in Chapter 6 of this thesis, it is evident that much of problem solving in complex development can equally benefit from a more efficient change in the structure of the developing teams rather than that of the whole organization. One conclusion is that enhancing IPT composition with experts from other interdependent subsystem teams would preempt a lot of problems at the early stages of the design, without requiring more drastic and costly changes to the entire organizational structure. This could be done by adding one representative from other subsystem teams with which there are tight interdependence, thus providing an extra expert set of eyes over design decisions affecting other interrelated subsystems in a tightly coupled architecture.

- Get the right answer at the right time: the key to problem solving is to call for the right help at the right time, and it was shown in this research that delays in asking for help turns small issues into bigger problems and feeds the firefighting phenomenon. The critical parts of this equation are therefore in 1) knowing who to ask for help (addressed in the second recommendation in this section) and 2) knowing when you actually need help. As the data analysis in the previous chapter showed, the vertical channel between subsystem IPT's and higher level program authority is critical for negotiation and mediation in problem solving, and so it naturally follows that this should be the first stop for advice on who and when to call for help. This was confirmed in the interviews where the reach of the "boss's network" was said to be a significant factor affecting the ability and timeliness of getting assistance.

- Support knowledge integration by formal incentives and measures: while knowledge is priceless, time is money and thus has a price, which makes it difficult for busy senior experts to share their valuable knowledge. In the course of this research and during a conference presentation, the author was asked how best to gain the attention of senior engineers who typically don't have time to spare for sharing lessons learned in formal sessions. My initial answer was that informal advice sharing was less time-consuming and just as effective of a

mechanism as formal share sessions and that it is underutilized in most organizations. However, as pointed to in the first recommendation above, the choice should not be an 'either-or' decision, and both formal and informal advice sharing should be enabled by the organization especially in complex environments where this research has shown that expert advice is critical for dealing with complexity. This can be supported by financial incentives and performance measures to encourage senior engineers to take some time out in order to share knowledge both within and across programs. The additional recommendation offered here is that functional groups are ideally positioned at the intersection of most knowledge flows and therefore have the unique opportunity to help identify and motivate senior experts to share advice both formally and informally, and there was no evidence in the data from this research of any incentives, direct or indirect, for senior staff to dedicate time for sharing advice or mentoring new hires. The mediation of functional groups in this case is less intrusive than that of moving experts which can cause the undesirable consequence of draining valuable knowledge from where it is needed, as previously discussed.

- Discredit existential myths: a fundamental dilemma facing knowledge integration across boundaries is how to avoid a zero-sum outcome, especially between separate organizational entities having distinct interests such as prime and supplier organizations, and it is already well established in previous research that there are many palpable and mutual benefits to be had from open knowledge transfer and sharing which add value for both parties, such as savings in time and cost during development. However, this research has further shown that knowledge integration in particular is inherently beneficial for both parties since opportunistic behavior is minimized during the stage of applying knowledge in collaborative problem solving. In other words, the potential for one party taking over the other's knowledge is unlikely to happen through knowledge integration since the interim learning is cumulative for both sides, meaning the prime is acquiring more design knowledge just as the supplier is acquiring more system integration knowledge, with potential benefits at the single organizational and collective network levels, namely through added system and subsystem capabilities. One such capability is the added flexibility in the supplier's subsystem design and the prime's system architecture, as illustrated in Figure 45 below (using an analogy from a common problem of having the correct adapter for plugging into different electric sockets).

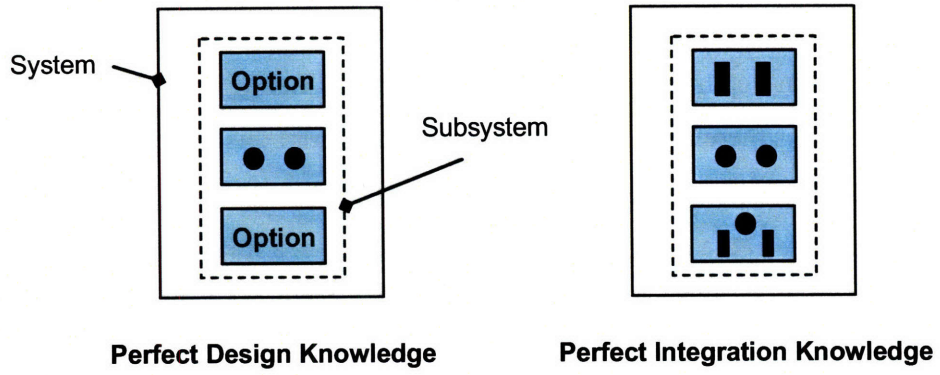


Figure 45: The Knowledge Integration Dichotomy

The left hand side of the figure shows that the more design knowledge the prime acquires, the more flexibility it can incorporate into the overall system to accommodate future design options. Conversely, the right hand side of the figure shows that the more system integration knowledge the supplier acquires, the more flexible its design will be since it will satisfy most any system configuration. It can therefore be argued that the production function for knowledge integration is such that prime and supplier knowledge are not perfectly substitutable and its global optimum is where the prime gives the supplier problems it can solve more easily, and the supplier gives the prime more complete solutions. Mutual benefit to some degree is therefore assured from a knowledge integration perspective, which is not necessarily the case when looking at knowledge transfer alone without the collaborative part of knowledge application during problem solving. As a result, organizations should discredit the myths about collaborative problem solving being a pitfall for losing knowledge to competitors and compromising competitive advantage.

7.4 Future Work

A deliberate strategy in this research was to explore the knowledge integration phenomenon in one of the most complex environments in terms of engineering complexity of the products (avionics) and the corresponding problems encountered. The choice of the defense context was also deliberate in order to investigate the knowledge integration and problem solving processes at the largest organizational network scale, and in order to explore the most challenging barriers

facing the integration of knowledge in any context. The investigation was further designed to be a comprehensive and in-depth analysis of different channels linking a variety of stakeholders at multiple levels, and it is therefore expected that the results from this research would be encompassing of other less complex, smaller-scale and more open-sharing contexts such as those encountered commonly in the commercial industry. However, a limitation of this research remains that it is only focused on the military aerospace industry, and therefore the first opportunity for future work is to extend the research into other large-scale complex engineering contexts in other industries in order to validate and build on the current insights in this thesis.

Another opportunity for future research is in framing the *critical mechanisms* for integrating *specific types of knowledge* and their impact on the problem solving process. The focus in this thesis was on the integration part (or the "how-to" part) of knowledge integration, in terms of describing the key integrative channels, strategies, practices and mechanisms for transferring, sharing and applying generic types of knowledge (e.g. tacit and explicit knowledge, design and integration knowledge) in different problem solving environments. However, an opportunity for future work remains, which is to explore in greater depth the knowledge part of knowledge integration (or the "know-what" part) so as to answer the questions: what mechanisms are critical for integrating a specific type of knowledge and what is their impact on problem solving efficiency and effectiveness? This would directly complement the current research by determining which types of mechanisms are appropriate for integrating specific types of knowledge. For example, in the case of complex avionics development, a specific type of required design knowledge might be about sensors technology, and an IPT responsible for radar development would be expected to have sufficient competence in this particular type of knowledge in order to solve radar design problems. The importance of this future work would then be in determining which types of mechanisms are critical for integrating specific types of knowledge an IPT needs in order to deal with specific types of problems. For example, if a radar subsystem IPT registers a 2 out of 5 level of expertise in radar signatures where this type of knowledge is critical for avoiding accuracy problems, and where this type of knowledge is mostly resident in the supplier base, it might be determined that an effective mechanism to overcome this shortcoming is to co-locate experts in radar signatures from the supplier during the integration phase of development. This investigation can be done at multiple levels of the

avionics suite for a particular subsystem, such as the component level, the mission subsystems level, and the air vehicle systems level, leading to a multi-scale conceptual scaffold describing knowledge integration in terms of both the "know-what" and the "how-to" parts of the process.

A more generalizable opportunity for future work from this research is in extending the current framework for knowledge integration from one describing individual strategies and mechanisms to a framing of dynamic organizational capabilities where each dynamic capability is a collection of complementary strategies and mechanisms appropriate for a particular context. The concept of organizational capability is an important yet largely undeveloped notion in the literature, and its value is in the strategic benefits it provides since dynamic capabilities are what differentiate organizations strategically and are intended to support the firm in sustaining its competitive advantage (Kogut and Zander 1995; Teece, Pisano et al. 1997; Zollo and Winter 2002). The main challenge in this future work would then be in tying specific capabilities to actual strategic-level benefits beyond savings in cost and schedule at the operational level.

Finally, a practical opportunity for future work is in applying some of the main findings from this research through the actual development of an information system for integrating problem solving experience and know-how in large-scale organizational environments. One example would be in the development of a software expert locator system which can track people's expertise in problem solving and allows searching for experts according to the specific problem solving knowledge they have accumulated, instead of by their academic or professional training or the types of positions they have held, all of which might not be as indicative of the person's competence in dealing with particular types of problems. Such a system would not only be more effective at identifying the right experts for the right problem than current state-of-the-art, but it would also be more dynamic and less costly to maintain than typical static systems in use at most organizations. This is because it would be possible to keep the system updated automatically as people engage in problem solving activities, and to have accurate and meaningful descriptions of their problem solving expertise, instead of having to rely on vague position titles and outdated skills based on previous training and education.

REFERENCES

Allen, T. (1997). Architecture and Communication Among Product Development Engineers, MIT Sloan School of Management.

Allen, T. J. (1977). Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information Within the R&D Organization, MIT Press.

Allen, T. J. (1997). "Managing Organizational Interfaces." FOOD AND DRUG LAW JOURNAL **52**: 173-178.

Allen, T. J. (2000). Organizational Structure for Product Development. MIT Sloan School of Management.

Almeida, P., J. Song, et al. (2002). "Are Firms Superior to Alliances and Markets? An Empirical Test of Cross-Border Knowledge Building." Organization Science **13**(2): 147-161.

Ancona, D. G. and D. F. Caldwell (1992). "Bridging the boundary: External activity and performance in organizational teams." Administrative Science Quarterly **37**(4): 605-633.

Aoshima, Y. (2002). "Transfer of System Knowledge Across Generations in New Product Development: Empirical Observations from Japanese Automobile Development." Industrial Relations **41**(4): 605-628.

Argote, L. (1999). Organizational learning : creating, retaining, and transferring knowledge. Boston, Kluwer Academic.

Argyres, N. S. (1999). "The impact of information technology on coordination: Evidence from the B-2 'Stealth'bomber." Organization Science **10**(2): 162-180.

Balaji, S. and M. K. Ahuja (2005). Critical Team-Level Success Factors of Offshore Outsourced Projects: A Knowledge Integration Perspective. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on System Sciences. Hawaii, IEEE Computer Society.

Baldwin, C. Y. and K. B. Clark (1997). "Managing in an Age of Modularity." Harvard Business Review.

Baldwin, C. Y. and K. B. Clark (2000). Design Rules: Volume 1. The Power of Modularity, MIT Press.

Bozdogan, K., J. Deyst, et al. (1998). "Architectural Innovation in Product Development through Early Supplier Integration." R&D Management **28**(3): 163-173.

Braha, D. and Y. Bar-Yam (2007). "The Topology of Large-Scale Engineering Problem-Solving Networks." Physical Review E **69**(1).

Brock and Schor (1990). Modular Avionics System Architecture (MASA) - the Impact of Fault Tolerance. IEEE Digitial Avionics Systems Conference.

Brown, J. S. and P. Duguid (1991). "Organizational learning and communities of practice: Toward a unified view of working, learning and innovation." Organization Science **2**(1, Special Issue: Organizational Learning: Papers in Honor of (and by) James G. March): 40-57. .

Brown, J. S. and P. Duguid (2001). "Knowledge and Organization: A Social-Practice Perspective." Organization Science **12**(2): 198-213.

Brown, S. L. and K. M. Eisenhardt (1995). "Product Development: Past Research, Present Findings, and Future Directions." The Academy of Management Review **20**(2): 343-378.

Browning, T. R. (1996). Systematic IPT Integration in Lean Development Programs. Engineering Systems Division. Cambridge, MIT.

Browning, T. R. (1997). Exploring Integrative Mechanisms with a View Towards Design for Integration. Fourth ISPE International Conference on Concurrent Engineering: Research and Applications.

Brusoni, S. and A. Prencipe (2001). "Managing Knowledge in Loosely Coupled Networks: Exploring the Links between Product and Knowledge Dynamics." Journal of Management Studies **38**(7): 1019-1035.

Brusoni, S. and A. Prencipe (2001). "Unpacking the Black Box of Modularity: Technologies, Products and Organizations." Industrial and Corporate Change **10**(1): 179-205.

Brusoni, S., A. Prencipe, et al. (2001). "Knowledge Specialization, Organizational Coupling, and the Boundaries of the Firm: Why Do Firms Know More Than They Make?" Administrative Science Quarterly **46**(4): 597-621.

Carlile, P. R. (2002). "A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development." Organization Science **13**(4): 442-455.

Carlile, P. R. (2004). "Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge Across Boundaries." Organization Science **15**(5): 555-568.

Carlile, P. R. and E. S. Rebentisch (2003). "Into the Black Box: The Knowledge Transformation Cycle." Management Science **49**(9): 1180-1195.

Chaltiel, P. Y., J. P. Gourion, et al. (1998). "Future Concepts for Airborne Electronic Warfare:" Fully Digital" Jamming And Missile Warning." MILITARY TECHNOLOGY **22**: 41-47.

Christensen, C., M. Verlinden, et al. (1999). Product Modularity, Vertical Disintegration and the Diffusion of Competence, Harvard Business School: 99-124.

Clark, K. B. and T. Fujimoto (1991). Product Development Performance: Strategy, Organization and Management in the World Auto Industry. Boston, Massachusetts, Harvard Business School Press.

Cohen, W. M. and D. A. Levinthal (1990). "Absorptive Capacity: A New Perspective on Learning and Innovation." Administrative Science Quarterly **35**(1, Special Issue: Technology, Organizations, and Innovation): 128-152. .

Committee on Aging Avionics in Military Aircraft (2001). Aging Avionics in Military Aircraft, NATIONAL ACADEMY PRESS.

Conner, K. R. and C. K. Prahalad (1996). "A Resource-Based Theory of the Firm: Knowledge versus Opportunism." Organization Science **7**(5): 477-501.

Crawley, E., O. de Weck, et al. (2004). "The Influence of Architecture in Engineering Systems." Engineering Systems Symposium.

Cross, R. and L. Sproull (2004). "More Than an Answer: Information Relationships for Actionable Knowledge." Organization Science **15**(4): 446–462.

Cusumano, M. and K. Nobeoka (1998). Thinking beyond lean : how multi-project management is transforming product development at Toyota and other companies. New York, Free Press.

Daft, R. L. and R. H. Lengel (1986). "Organizational Information Requirements, Media Richness and Structural Design." Management Science **32**(5): 554-571.

Davenport, T. and L. Prusak (1998). Working Knowledge: How Organizations Manage What They Know, Harvard Business School Press.

De Boer, M., F. A. J. Van Den Bosch, et al. (1999). "Managing Organizational Knowledge Integration in the Emerging Multimedia Complex." Journal of Management Studies **36**(3): 379-398.

Demsetz, H. (1988). "The Theory of the Firm Revisited." Journal of Law, Economics, & Organization **4**(1): 141-161.

Dosi, G., M. Hobday, et al. (2000). Problem-Solving Behaviors, Organisational Forms and the Complexity of Tasks. Laboratory of Economics and Management Working Paper Series. Pisa, Italy, Sant'Anna School of Advanced Studies.

Dyer, J. H. and W. Chu (2003). "The Role of Trustworthiness in Reducing Transaction Costs and Improving Performance: Empirical Evidence from the United States, Japan, and Korea." Organization Science **14**(1): 57–68.

Dyer, J. H. and K. Nobeoka (2000). "Creating and Managing a High-Performance Knowledge-Sharing Network: The Toyota Case." Strategic Management Journal **21**(3): 345-367.

Eisenhardt, K. M. and B. N. Tabrizi (1995). "Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry." Administrative Science Quarterly **40**(1): 84-110.

Eppinger, S. and R. Gulati (1996). The Coupling of Product Architecture and Organizational Structure Decisions, MIT Sloan School of Management Working Paper.

Eppinger, S. D. (2002). Patterns of Product Development Interactions. Working Paper ESD-WP-2003-01.05-ESD Internal Symposium, Massachusetts Institute of Technology - Engineering Systems Division: 8.

Fabian and Rayl (1998). "Advanced Avionics System Architecture." AIAA/IEEE/SAE **2**.

Filmer, B. (2003). "Open Systems Avionics Architectures Considerations." IEEE AES Systems Magazine(September 2003).

Fine, C. H. (2000). "Clockspeed-Based Strategies for Supply Chain Design." Production and Operations Management **9**(3).

Fine, C. H. and D. E. Whitney (1996). "Is the make-buy decision process a core competence?" Paper submitted to MIT IMVP Sponsors' Meeting, at Sao Paulo, Brazil.

Foss, N. J. (1993). "Theories of the firm: contractual and competence perspectives." Journal of Evolutionary Economics **3**(2): 18.

Foss, N. J. (1996a). "Knowledge-Based Approaches to the Theory of the Firm: Some Critical Comments " Organization Science **7**(5): 470-476

Fujimoto, T. (1999). The Evolution of a Manufacturing System at Toyota. New York, Oxford University Press.

Fujimoto, T. (2002). Architecture, Capability and Competitiveness of Firms and Industries, University of Tokyo.

Galbraith, J. R. (1974). Organizational Design: An Information Processing View, M. Wiener.

Galvin, P. and A. Morkel (2001). "The Effect of Product Modularity on Industry Structure: The Case of the World Bicycle Industry." Industry and Innovation **8**(1): 31-47.

Glaser, B. G. and A. L. Strauss (1967). The discovery of grounded theory; strategies for qualitative research Chicago, Aldine Pub. Co.

Gomes-Casseres, B. (1994). "Group versus group: How alliance networks compete." Harvard Business Review **72**(4): 62-74.

Gourlay, S. (2006). "Towards conceptual clarity for 'tacit knowledge': a review of empirical studies." Knowledge Management Research & Practice **4**: 60–69.

Grant, R. M. (1996a). "Toward a Knowledge-Based Theory of the Firm." Strategic Management Journal **17**(Winter Special Issue): 109-122.

Grant, R. M. (1996b). "Prospering in Dynamically Competitive Environments: Organizational Capability as Knowledge Integration." Organization Science **7**(4): 375-387.

Grant, R. M. and C. Baden-Fuller (1995). "A knowledge-based theory of inter-firm collaboration." Academy of Management **Best Paper Proceedings**: 17-21.

Gupta, A. K. and V. Govindarajan (2000). "Knowledge flows within multinational corporations." Strategic Management Journal **21**(4): 473-496.

Hansen, M. T. (2002). "Knowledge Networks: Explaining Effective Knowledge Sharing in Multiunit Companies." Organization Science **13**(3): 232-248.

Hansen, M. T., N. Nohria, et al. (1999). "What's your Strategy for Managing Knowledge." Harvard Business Review: 106-116.

Hedlund, G. (1994). "A Model of Knowledge Management and the N-Form Corporation." Strategic Management Journal **15**: 73-90.

Henderson, R. and K. Clark (1990). "Architectural Innovation: The Reconfiguration Of Existing Product Technologies and the Failure of Established Firms." Administrative Science Quarterly **35**(1): 9-30.

Hicks, B. (2004). Transforming Avionics Architectures to Support Network Centric Warfare. Digital Avionics Systems Conference.

Hoopes, D. G. and S. Postrel (1999). "Shared Knowledge, "Glitches," and Product Development Performance." Strategic Management Journal **20**(9): 837-865.

Imbesi, D. J. and W. K. Kaplow (1992). "The Pave Pace Integrated Core Processor." Aerospace and Electronics Conference, 1992. NAECON 1992., Proceedings of the IEEE 1992 National: 806-814.

Johnson, J. D., W. A. Donohue, et al. (1994). "Differences Between Formal and Informal Communication Channels." Journal of Business Communication **31**: 111-122.

Joint Advanced Strike Technology Program (1994). Avionics Architecture Definition, Version 1.0.

Jonassen, D. H. (2000). "Toward a Design Theory of Problem Solving." Educational Technology Research and Development **48**(4): 63-85.

Kogut, B. and U. Zander (1992). "Knowledge of the Firm, Combinative Capabilities, and the Replication of Technology." Organization Science **3 Focused Issue: Management of Technology**(3): 383-397.

Kogut, B. and U. Zander (1995). "Knowledge and the speed of the transfer and imitation of organizational capabilities: an empirical test." Organization Science **6**(1): 76-92.

Kogut, B. and U. Zander (1996). "What Firms Do? Coordination, Identity, and Learning." Organization Science **7**(5): 502-518.

Langlois, R. N. (2002). "Modularity in Technology and Organization." Journal of Economic Behavior & Organization **49**: 19-37.

Larsson, R., L. Bengtsson, et al. (1998). "The interorganizational learning dilemma: Collective knowledge development in strategic alliances." Organization Science **9**(3): 285-305.

Loewy, R. (1999). Avionics: A "New" Senior Partner in Aeronautics. 37th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada.

Maier, M. W. (1996). "Systems architecting: an emergent discipline?" Aerospace Applications Conference, 1996. Proceedings., 1996 IEEE **3**.

Malone, T. W. and K. Crowston (1991). "Toward an interdisciplinary theory of coordination."

Malone, T. W. and K. Crowston (1994). "The interdisciplinary study of coordination." ACM Computing Surveys (CSUR) **26**(1): 87-119.

March, J. G. (1991). "Exploration and Exploitation in Organizational Learning." Organization Science **2**(1, Special Issue: Organizational Learning: Papers in Honor of (and by) James G. March): 71-87.

Mayer, R. E. and M. C. Wittrock (1996). "Problem-solving transfer." Handbook of educational psychology: 47-62.

Miles, M. B. and A. M. Huberman (1994). Qualitative Data Analysis: An Expanded Sourcebook, Sage Publications.

Moir, I. and A. Seabridge (2001). Aircraft Systems: Mechanical, Electrical, and Avionics Subsystems Integration. Reston, VA, Professional Engineering Pub.

Moir, I. and A. Seabridge (2004). Design and Development of Aircraft Systems: An Introduction. Reston, VA, Professional Engienering Pub.

Moir, I. and A. Seabridge (2006). Military Avionics Systems. Chichester, England John Wiley & Sons.

Morgan, D. R. (1995). Military avionics twenty years in the future. AIAA / IEEE Digital Avionics Systems Conference.

Moses, J. (2002). Complexity and Flexibility, MIT.

Mumford, E. (1998). Problems, Knowledge, Solutions: Solving Complex Problems. International Conference on Information Systems. Helsinki, Finland, Association for Information Systems.

Newell, A. and H. A. Simon (1972). Human problem solving, Prentice-Hall.

Nickerson, J. A. and T. R. Zenger (2004). "A Knowledge-Based Theory of the Firm - The Problem-Solving Perspective." Organization Science **15**(6): 15.

Nickols, F. (1994). "Reengineering the Problem Solving Process." Performance Improvement Quarterly **7**(4).

Nobeoka, K. (1993). Multi-project management: strategy and organization in automobile product development, Massachusetts Institute of Technology, Sloan School of Management.

Nobeoka, K. and M. Cusumano (1994). Multi-Project Strategy and Market-Share Growth: The Benefits of Rapid Design Transfer in New Product Development, IMVP Working Papers. Cambridge, MA: The MIT Press.

Nonaka, I. (1994). "A Dynamic Theory of Organizational Knowledge Creation." Organization Science **5**(1): 14-37.

Nonaka, I. and H. Takeuchi (1995). The knowledge-creating company : how Japanese companies create the dynamics of innovation. New York, Oxford University Press.

Nonaka, I. and D. Teece (2001). Managing industrial knowledge : creation, transfer and utilization. London ; Thousand Oaks, Calif., SAGE.

Novak, S. and S. D. Eppinger (2001). "Sourcing by Design: Product Complexity and the Supply Chain." Management Science **47**(1): 189-204.

Okhuysen, G. A. and K. M. Eisenhardt (2002). "Integrating Knowledge in Groups: How Formal Interventions Enable Flexibility." Organization Science **13**(4): 370-386.

Pandit, N. R. (1996). "The Creation of Theory: A Recent Application of the Grounded Theory Method." The Qualitative Report **2**(4): 1-15.

Parnas, D. (1972). "On the Criteria for Decomposing Systems into Modules." Communications of the ACM **15**(12): 1053-1058.

Penrose, E. (1955). "Limits to the Growth and Size of Firms." The American Economic Review **45**(2): 531-543.

Polanyi, M. (1966). The Tacit Dimension, Routledge & Kegan Paul Ltd.

Postrel, S. (2002). "Islands of Shared Knowledge: Specialization and Mutual Understanding in Problem-Solving Teams." Organization Science **13**(3): 303–320.

Prahalad, C. K. and G. Hamel (1990). "The Core Competence of the Corporation." Harvard Business Review **68**(3): 79-91.

Prencipe, A. (1997). "Technological competencies and product's evolutionary dynamics a case study from the aero-engine industry." Research Policy **25**(8): 1261-1276.

Prencipe, A. (2000). "Breadth and depth of technological capabilities in CoPS: the case of the aircraft engine control system." Research Policy **29**(7): 895-911.

Prusak, L. (1996). "The Knowledge Advantage." Strategy & Leadership **24**(2): 6-8.

Purvis, R. L., V. Sambamurthy, et al. (2001). "The Assimilation of Knowledge Platforms in Organizations: An Empirical Investigation." Organization Science **12**(2): 117- 135.

Quaranta, P. (2000). "The evolution of Avionic System Architectures." Military Technology(Oct. 2000): pp. 86-89.

Quaranta, P. (2002). "Modern Sensors Packages for Combat Aircraft." Military Technology **2**: 92-93.

Rechtin, E. (1991). Systems Architecting: Creating and Building Complex Systems, Prentice Hall.

Repenning, N. (2001). "Understanding fire fighting in new product development." The Journal of Product Innovation Management. **18**(5).

Robertson, D. and K. Ulrich (1998). "Planning for product platforms." Sloan Management Review **39**(4): 19-31.

Robertson, P. L. and R. N. Langlois (1995). "Innovation, networks, and vertical integration." Research Policy **24**: 543-562.

Robson, C. (2002). Real World Research–Second edition, Blackwell Publishers Ltd., Oxford, UK.

Sako, M. (2003). Modularity and Outsourcing: The Nature of Co-Evolution of Product Architecture and Organisation Architecture in the Global Automotive Industry. Eleventh GERPISA International Colloquium. Paris, France.

Sanchez, R. (1997). "Preparing for an Uncertain Future: Managing Organizations for Strategic Flexibility." International Studies of Management & Organization **27**(2).

Sanchez, R. and J. T. Mahoney (1996). "Modularity, flexibility and knowledge management in product and organization design." Strategic Management Journal **17**(Winter special issue): 63-76.

Sanderson, S. and M. Uzumeri (1995). "Managing product families: The case of the Sony Walkman." Research Policy **24**(5): 761-782.

Santos, F. M. and K. M. Eisenhardt (2005). "Organizational Boundaries and Theories of Organization." Organization Science **16**(5): 491-508.

Senge, P. (1994). The Fifth Discipline: The Art and Practice of the Learning Organization, Currency.

Sheard, S. A. and E. M. Margolis (1995). Team Structures for Systems Engineering in an IPT Environment. Proceedings of the Fifth Annual International Symposium of INCOSE. St. Louis.

Simon, H. (1983). "Search and reasoning in problem solving." Artificial Intelligence **21**(1): 7-29.

Simon, H. A. (1962). "The Architecture of Complexity." Proceedings of the American Philosophical Society **106**(6): 467-482.

Simon, H. A. (1973). "Applying Information Technology to Organization Design." Public Administration Review **33**(3): 268-278.

Simon, H. A. (1973). "The Structure of Ill Structured Problems." Artificial Intelligence **4**(3): 181-201.

Simon, H. A. (1976). "How Complex are Complex Systems?" PSA: Proceedings of the Biennial Meeting of the Philosophy of Science Association **1976**: 507-522.

Sosa, M. E. and S. D. Eppinger (2002). "Factors That Influence Technical Communication in Distributed Product Development: An Empirical Study in the Telecommunications Industry." IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT **49**(1): 45-58.

Sosa, M. E., S. D. Eppinger, et al. (2000a). Designing Modular and Integrative Systems. ASME 2000 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, Maryland, ASME.

Sosa, M. E., S. D. Eppinger, et al. (2000b). Understanding the Effects of Product Architecture on Technical Communication in Product Develoment Organizations, Massachusetts Institute of Technology, Sloan School of Management.

Sosa, M. E., S. D. Eppinger, et al. (2003). "Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions." Journal of Mechanical Design **125**: 240.

Sosa, M. E., S. D. Eppinger, et al. (2004). "The Misalignment of Product Architecture and Organizational Structure in Complex Product Development." Management Science **50**(12): 1674-1689.

Spender, J.-C. (1996). "Making Knowledge the Basis of a Dynamic Theory of the Firm." Strategic Management Journal **17**(Special Issue: Knowledge and the Firm. (Winter, 1996)): 45-62.

Spender, J. C. (1996). "Competitive advantage from tacit knowledge? Unpacking the concept and its strategic implications." Organizational Learning and Competitive Advantage: 56-73.

Spitzer, C. (1987). Digital avionics systems. Englewood Cliffs, N.J., Prentice-Hall.

Spitzer, C. R. (2001). The Avionics Handbook, CRC Press.

Star, S. L. (1989). The structure of ill-structured solutions: Boundary objects and heterogeneous distributed problem solving. Menlo Park, CA., M. Huhns and L. Gasser, eds.

Strauss, A. and J. Corbin (1990). Basics of qualitative research : grounded theory procedures and techniques. Newbury Park, California, Sage Publications.

Strauss, A. L. (1987). Qualitative Analysis for Social Scientists, Cambridge University Press.

Szulanski, G. (1996). "Exploring internal stickiness: Impediments to the transfer of best practice within the firm." Strategic Management Journal **17**(Winter 1996): 27-44.

Takeishi, A. (2001). "Bridging inter- and intra-firm boundaries: management of supplier involvement in automobile product development." Strategic Management Journal **22**(5): 403-437.

Takeishi, A. (2002). "Knowledge Partitioning in the Interfirm Division of Labor: The Case of Automotive Product Development." Organization Science **13**(3): 321-338.

Takeuchi, H. and I. Nonaka (2004). Hitotsubashi on knowledge management, John Wiley & Sons (Asia) Singapore.

Teece, D., G. Pisano, et al. (1997). "Dynamic Capabilities and Strategic Management." Strategic Management Journal (1986-1998). **18**(7): 509-533.

Ulrich, K. (1995). "The role of product architecture in the manufacturing firm." Research Policy **24**: 419-441.

Utterback, J. (1996). Mastering the Dynamics of Innovation. Boston, MA, Harvard Business School Press.

Vincenti, W. G. (1990). What Engineers Know and how They Know it: Analytical Studies from Aeronautical History, Johns Hopkins University Press.

Von Hippel, E. (1994). ""Sticky Information" and the Locus of Problem Solving: Implications for Innovation." Management Science **40**(4): 429-439.

Walz, D. B., J. J. Elam, et al. (1993). "Inside a software design team: A look at knowledge acquisition, sharing, and integration activities." Communications of the ACM **36**(10): 63-74.

Whitney, D. E. (2004). Mechanical Assemblies: Their Design, Manufacture, and Role in Product Development, Oxford University Press.

Wissmann, L. A. and A. A. Yassine (2005). Product Architecture and the Firm, PD-Lab Working Paper, 2004,(1).

Wolfe, C. A., M. E. Campbell, et al. (1996). "Integrated CNI avionics using F-22 modular products." Aerospace and Electronics Conference, 1996. NAECON 1996., Proceedings of the IEEE 1996 National **1**.

Womack, J. and D. Jones (1996). "Lean Thinking: Banish Waste and Create Wealth in Your Organisation." Rawson Associates, New York.

Womack, J. P., D. T. Jones, et al. (1991). The Machine that Changed the World: How Japan's Secret Weapon in the Global Auto Wars Will Revolutionize Western Industry, HarperPerennial.

Yang, J. (2005). "Knowledge integration and innovation: Securing new product advantage in high technology industry." Journal of High Technology Management Research **16**(1): 121-35.

Zollo, M. and S. G. Winter (2002). "Deliberate Learning and the Evolution of Dynamic Capabilities." Organization Science **13**(3): 339.

APPENDIX A: FIELD INSTRUMENTS

This appendix provides an overview of the exploratory and semi-structured interview questions used in the first three rounds of field research in each case study, and a template of the structured questionnaire administered in the fourth (final) round of the research.

A.1 Sample Exploratory Interview Questions:

(1) What is your background and relationship to the case study? State your program affiliation and any association you have with other aircraft programs than your own.

(2) What are your organization's most recent (or major) knowledge transfer and knowledge sharing initiatives?

(3) What are the knowledge transfer and sharing mechanisms most emphasized in your organization?

(4) What are the main advantages and disadvantages of your organization's structural arrangement for knowledge transfer and sharing with other programs and functions?

(5) Is the supporting infrastructure (IT, communities of practice, integrated teams, etc.) effective in transferring knowledge necessary for problem solving? If not, what are the barriers impeding the efficient transfer of knowledge through the infrastructure?

(6) Describe formal strategies, practices or mechanisms established in your program or at the enterprise-level that enable the transfer and/or sharing of engineering knowledge (about system architectures, technologies, processes, etc…) with other programs and functions.

(7) Describe informal strategies, practices or mechanisms used for the same purpose as in question (4) above.

(8) Are there clearly established and communicated goals for knowledge transfer and/or sharing in your program? If so, what are the main objectives? If not, what are the impediments?

(9) Do the established knowledge transfer and sharing practices effectively support problem solving? If yes, what are the most effective practices? If not, what are the impediments?

(10) Are there incentives in place to foster knowledge transfer and sharing between the prime organization and major suppliers? If so, can you provide examples?

(11) Are "lessons learned" effectively shared in your organization? If so, what are the most effective mechanisms for sharing lessons learned? If not, what are the main barriers impeding the sharing of lessons learned?

(12) When and how are key suppliers integrated into the design and development process? Are key suppliers represented on the integrated product teams in the individual programs? Does the involvement of suppliers involve special contractual arrangements, such as proprietary restrictions?

A.2 Sample Focused Interview Questions:

(13) How many years of experience do you have with the company, with your current program or division, and with your current IPT? What is your current position and role in developing the subject avionics system?

(14) How is the responsibility for developing your subsystem divided up across team, program, functional and organizational boundaries? What are the most common types of teams in your organization (e.g. functional, cross-functional, product teams)?

(15) What other organizational groups/entities do you consult most frequently to solve major technical problems, both formally and informally?

(16) What type(s) of major technical problems do you encounter most frequently with the development of your particular subsystem?

(17) Describe areas/instances of team commonality (e.g. integrated teams) with other aircraft programs.

(18) Describe areas/instances of rivalry (e.g. constructive or negative competition) between your program and other aircraft programs in the enterprise, as well as their causes and drawbacks on the problem solving process.

(19) What mechanisms are used to exchange engineering knowledge about subsystem designs and technologies (e.g. specifications, standards) across programs, across engineering functions, across generations of components, and with suppliers?

(20) What mechanisms are used to exchange relational knowledge about system architectures (e.g. system integration rules) across programs, across engineering functions, across generations of systems, and with suppliers?

(21) What is the role of subsystem engineers, if any, in supporting system integration efforts? What is the role of system integrators, if any, in supporting subsystem design efforts?

(22) Name the most effective knowledge transfer or sharing practice (within or across programs or with suppliers) that supports design commonality for multiple platforms?

(23) Draw a network map identifying links and points-of-contact (POC's) for sharing program knowledge with other aircraft programs and other parts of the enterprise.

(24) Can you identify gaps in the network map impeding effective transfer and sharing of program knowledge across program boundaries?

A3: Structured Questionnaire

PROBLEM DESCRIPTION: *Describe a major technical problem you encountered with your subsystem*

PROBLEM CATEGORY: *Please indicate the problem type (or root cause)*

MISSION CRITICAL / SAFETY CRITICAL:
SYSTEM DESIGN / SYSTEM INTEGRATION, H/W and/or S/W :
OTHER:

PROBLEM CHARACTERISTICS: *Please describe the technical characteristics of the problem*

SCALE (**Few** Subsystems Affected / **Many** Subsystems Affected):
SCOPE (**Internal** / **External** involving Prime or Other Suppliers):
SEVERITY (**Single**-System Disruption / **Multi**-System Disruption):
NOVELTY (**New** Problem or Technology / **Old** Problem or Mature Technology):

PROBLEM SOLVING APPROACH: *Describe how the problem was identified and/or solved*

xxx description of problem solving process here xxx

Indicate with an (**X**) *below which knowledge exchange mode was used most frequently:*

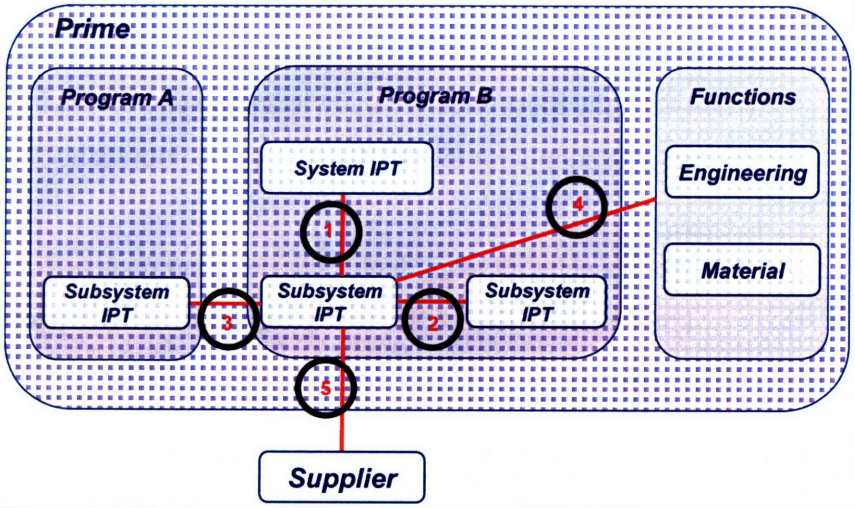
Exchanged Information (e.g. transfer of system requirements / specs…): **X**
Exchanged Advice (e.g. Red Team suggestions): **X**
Exchanged Assistance (e.g. Tiger Team direct involvement): **X**

KNOWLEDGE INTEGRATION: *Refer to the figure and indicate which channels and mechanisms were used to exchange information, advice and assistance in problem solving*

CHANNEL(S): CIRCLE 1, 2, 3, 4, 5

MECHANISM(S):

Requirements, specifications
Brainstorming meetings
Special action teams (tiger team, taskforce)
Special review teams (red/gold team…)
Lessons learned sessions
Shared databases
Design reviews
Experts (SME, Graybeard…)
Co-location, Site visits
Liaison devices (Liaison engineer…)
Other:



APPENDIX B: SUMMARY TABLES FROM INTERVIEW DATA REDUCTION

For quick reference, the tables in this appendix provide a summarized alphabetical listing of the most emphasized characteristics of the knowledge integration process (referred to as "critical instances" in § 6.1.1) as identified in the reduction of the interview data from all case studies.

Table 41: Intra-Program Knowledge Integration

Channels	Strategies / Practices	Mechanisms
1. Intra – Program	1. Apprenticeship 2. Calling on program experts 3. Communication 4. Conferences 5. Integrated concurrent engineering 6. Joint problem solving with program experts 7. Joint problem solving by special teams 8. Knowledge capture and reuse 9. Periodic reviews 10. Problem solving in IPT 11. Sharing lessons learned 12. Status updates 13. Team structuring 14. Training (general) by program	1. Boundary objects (prototypes, models, simulations) 2. Brainstorming meetings (team problem solving) 3. Brown bag sessions (informal ad-hoc share sessions) 4. Buddy system (social networking) 5. Design and zone reviews (systems readiness review SRR, preliminary design review PDR, critical design review CDR) 6. Work documents (memos, manuals, reports, contracts) 7. Information tools (emails, teleconference calls, netmeetings) 8. "Gurus" (senior experts, not involved in problem solving) 9. "Wizards" (experts in root-cause analysis, lab testing) 10. "Subject matter experts"(experts in a particular system) 11. "Tech fellows (discipline specific experts) 12. Integrated work tools (for concurrent modeling, drafting, design, workflow management) 13. IPT structure to co-locate multiple disciplines 14. Lessons learned databases 15. Program data libraries (electronic repositories for programmatic information, software code) 16. Program familiarization 17. Program templates/checklists 18. Share sessions (face-to-face or teleconference meetings) 19. Special action teams (tiger teams, taskforces) 20. Special review teams (gold teams, red teams) 21. Mentoring (technical apprenticeship) 22. Monthly knowledge symposium 23. Status meetings (presentations for updates) 24. Troubleshooting reviews (root-cause analysis sessions) 25. Walk-and-talk (chatting with peers to ask questions) 26. Work documents (manuals, reports, logs, test data, system architecture charts, requirements/specification documents)

Table 42: Program-to-Program Knowledge Integration

Channels	Strategies / Practices	Mechanisms
2. Program – Program	1. Communication 2. Design, technology and process commonality 3. Joint problem solving with other program experts 4. Joint problem solving by special teams 5. Periodic reviews 6. Professional affinity groups 7. Sharing information resources 8. Sharing lessons learned 9. Status updates	1. Call-up of counterparts/colleagues in other programs 2. Chief engineers periodic meetings 3. Common suppliers across programs 4. Communities of practice 5. Focus groups (systems engineering focus group) 6. Gurus from one program share lessons with another program 7. Loan out subject matter experts, tech fellows and gurus to another program 8. Mixed-program meetings 9. Multi-program lessons learned database 10. Multi-program share sessions 11. Non-advocate design reviews by one program for another 12. Participation in special action teams (tiger teams, taskforces) 13. Participation in special review teams (gold teams, red teams) 14. Program managers/chief engineers/team leads from older programs convene to share lessons at start of new program 15. Prototyping new system designs, architectures and technologies by older programs for future ones 16. Shared repositories (common databases, shared data libraries, intranets) 17. Sharing new design and architectural concepts 18. Sharing new technologies, engineering and manufacturing discoveries and materials 19. Sharing processes and procedures 20. White papers 21. Working groups (avionics working group)

Table 43: Function-to-Program Knowledge Integration

Channels	Strategies / Practices	Mechanisms
3. Function – Program	1. Calling on function experts 2. Career and skills development 3. Joint problem solving with function experts 4. Joint problem solving with function engineers 5. Joint problem solving by special teams 6. Knowledge capture and dissemination 7. Periodic reviews 8. Sharing lessons learned 9. Supporting cross-boundary communication and joint problem solving 10. Supporting integrated concurrent engineering 11. Supporting cross-program commonality 12. Supporting cross-program communication 13. Supporting cross-program standardization 14. Supporting process improvement 15. Training (specialized) by function	1. Capture and dissemination of lessons learned across programs (checklists for typical problems) 2. Capture and sharing of best practices across programs 3. Commonality initiatives (common designs, architectures) 4. Continuous improvement (dedicated office for transfer of information about process improvements between programs). 5. Developing and dissemination of technical documentation (software development manual, systems engineering guide) 6. Deploying integrated work tools (integrated management tools, common product design and management systems) 7. Deploying integrated information systems (expertise locator systems, intranets, shared databases) 8. Forming and participating in special action teams (tiger teams, taskforces) 9. Forming and participating in special review teams (gold teams, red teams) 10. "Graybeards" (senior experts not deployed to programs) 11. Independent design reviews / walkthroughs 12. Information audits (of standards, processes, procedures) 13. Initiatives for summarizing information 14. Integrating dispersed competencies and skills (from handshake process to close coordination) 15. Job rotation (personnel moved around functions and programs to gain broad skills) 16. Liaison engineers (spanning group boundaries) 17. Loaning out experts (tech fellows) to programs 18. Long-term assignments (hire and deploy staff to programs) 19. Material integrators (spanning organizational boundaries) 20. On-boarding (training and familiarization for new hires with tools, processes, acronyms) 21. Short-term assignments (deploying function personnel to help programs with problem solving) 22. Program reviews (capability reviews, team accountability reviews) 23. Sharing generic info across programs (supplier info) 24. Standardization of processes, procedures across programs (standard software development process, design templates) 25. Technical courses (design courses taught across programs)

Table 44: Prime-to-Supplier Knowledge Integration

Channels	Strategies / Practices	Mechanisms
4. Prime – Supplier	1. Communication 2. Conferences 3. Design, technology and process transfer 4. Early supplier integration 5. Joint problem solving with supplier experts 6. Joint problem solving by team reviews 7. Joint problem solving by special teams 8. Problem solving meetings 9. Process improvement 10. Sharing information resources 11. Sharing lessons learned 12. Single sourcing 13. Supplier management 14. Status updates 15. Tight partnerships	1. Ad-hoc sharing of lessons learned in meetings, discussions and reviews 2. Annual supplier conventions, monthly symposiums 3. Brokering supplier marriages 4. Buyer Furnished Equipment (BFE) 5. Carrot-and-stick management strategy 6. Co-location (long-term) of prime and supplier personnel 7. Concurrent design/management tools (integrated computer aided engineering tools, common risk management tool) 8. Decision boards (technical decision board grouping prime and supplier experts) 9. Design or module commonality across programs 10. Electronic and paper communication (emails, net-meetings, memos, contracts, technical agreements, teleconference calls) 11. Joint product assessment teams (forum grouping prime and supplier experts to share ideas) 12. Long term agreements with suppliers 13. Mergers with suppliers/competitors 14. Mixed prime-supplier participation in special action teams (red teams, tiger teams, taskforces) 15. Rotation of prime or supplier experts (short-term) 16. Site visits (short-term) by prime or supplier personnel 17. Shared databases (requirements management database, common problem reporting system, risk database, product data management system) 18. Special communication forums (strategic supplier advisory group for sharing lessons learned) 19. Strategic supplier working groups (for working out supplier problems across programs) 20. Supplier integrated product development 21. Technical meetings at prime's location 22. Technical interchange meetings at supplier location 23. Toyota model of vendor village 24. Transfer of best practices to/from supplier 25. Using the same supplier across programs 26. Work documents flow (requirements documents, interface control documents, drawings, failure reports, engineering change requests/orders, design evaluation/change requests, system architecture charts, testing documents)

Table 45: Knowledge Integration along Other Channels

Channels	Strategies / Practices	Mechanisms
5. Corporate – Program	1. Independent research and development (IRAD) 2. Organizational structuring 3. Periodic reviews 4. Process improvement	1. R&D transfer to programs 2. Program-oriented organizational structure to integrate across functional disciplines 3. Function-oriented organizational structure to integrate across programs 4. Program management council 5. Monthly program reviews by front office to share lessons learned
6. Customer – Prime	5. Customer and supplier management 6. Periodic reviews 7. Problem solving with customer 8. Status updates 9. Joint problem solving by special teams	6. Program and product team leadership mediate between customer and supplier 7. Communication to transfer feedback to/from customer 8. Requirements brainstorming meetings with customer 9. Independent reviews by customer 10. Taskforce participation by customer experts
7. Industry – Prime	10. Joint problem solving by special teams 11. Design, technology and process transfer	11. Taskforce participation by academia, industry consultants/experts 12. Systems engineers monitor/roadmap/transfer industry innovations
8. Supplier – Supplier	12. Design, technology and process transfer 13. Status updates 14. Joint problem solving by boundary spanners	13. Meetings, teleconferences, communication for updates 14. Supplier working groups (interface contractor working group) 15. Prime mediated/facilitated problem solving between suppliers 16. Transfer of design information, buyer furnished equipment

Table 46: Enabling Conditions for Knowledge Integration

Channels	Enabler Instances
1. Intra – Program	1. Good working environment (open sharing inside program) 2. Retaining core competencies in-house (experience helps with problem solving) 3. Codifying the learning of retiring senior engineers so it's repeatable 4. Social networks between peers (open communication among peers) 5. Financial Incentives (awards to recognize problem solving, firefighting and innovation) 6. Maturity of program, teams, infrastructure (stable environment, well established relations) 7. Maturity of individuals (experience of senior versus junior engineers) 8. Shared identity (one program as one team)
2. Program – Program	9. Financial incentives (corporate performance incentives instead of program performance) 10. Social networks (informal people relationships to overcome confidentiality barriers) 11. Shared fate (organizational culture as the glue between programs) 12. Mobilization capability (can rally troops for major problems, overcome program firewalls)
3. Function – Program	13. Use of metrics (for training, tool use) 14. Financial resources (large training budget) 15. Checks and balances (as process owners functions follow up on action items) 16. Feedback and verification (death by reviews, scorecards) 17. Financial Incentives (award for best technical paper) 18. Trust relationship between functions and programs (function is neutral negotiator)
4. Prime – Supplier	19. Easy access for suppliers to prime's knowledge resources 20. Deep visibility for suppliers into prime's knowledge resources 21. Liberal knowledge sharing policy for processes, programmatic information (operating concepts) and administrative (non-technical) information 22. Open sharing mentality (knowledge sharing considered more critical than proprietary protection) 23. Financial Incentives (reward suppliers for team performance so suppliers help each other) 24. Shared goals (no competition for work)
5. Corporate – Program	25. Succession planning for retiring leaders 26. Leadership support for giving engineering more free time/breaks to form social networks
6. Customer – Prime	27. Shared goals (customer willing to forego specs in favor of performance-based solutions)
7. Industry – Prime	28. Industry standardization (same industry standards pulling prime and supplier together)
8. Supplier – Supplier	29. Shared goals (no competition for work)

Table 47: Barriers to Knowledge Integration

Channels	Barrier Instances
1. Intra – Program	1. Brain drain (from program and organization) 2. Classified information (technical information, engineering and manufacturing discoveries) 3. Firefighting culture (schedule pressure so no time to get information, no follow-through on expert suggestions, concurrent multiple developments drain knowledge resources) 4. Hierarchy barriers (reluctance to "air dirty laundry" about problems when middle management in meetings, and "one man's agenda" by managers makes it difficult to share knowledge about other problems) 5. Knowledge divide (disconnect between old knowledge of boss/gurus and new knowledge of junior personnel – knowledge of today different than yesterday's, cultural disconnects between old and young adds to knowledge gap between generations) 6. Lack of formal incentives (informal personal initiatives to share knowledge when possible) 7. Personal barriers (personality differences/incompatibilities) 8. Poor planning (concurrent engineering by accident not by planning) 9. Pride in problem solving (reluctance of engineers to seek help) 10. Team structure limitations (tendency of IPT's to work in isolation, rivalry with other program IPT's due to their self-sufficiency)
2. Program – Program	11. Classified information restrictions (can't share technical knowledge outside program) 12. Custom customer needs (different customer requirements are a barrier to commonality, can't share architecture and complete design) 13. Custom program needs (program processes/procedures cannot always be standardized because of different needs/contexts, tribal knowledge) 14. Development dependence risks (common development creates added risk of reliance on the outside that programs don't want to assume) 15. Firefighting culture (small attendance at multi-program share sessions since senior engineers don't have time to share knowledge, junior engineers don't have time to attend) 16. Lack of formal process (burden falls on programs to share/acquire knowledge across boundaries) 17. Knowledge stealing (new programs cherry pick from old programs, stealing/robbing/draining limited knowledge resources instead of knowledge sharing) 18. Limited shared resources (competition between programs for the same supplier expert's time) 19. Organizational structure limitations (tendency of program-based structure to disconnect programs from outside knowledge in other programs in favor of internal integration across functional disciplines, project groups become divided) 20. Proprietary restrictions (can't share proprietary partner knowledge if partner on one program is a competitor on another program) 21. Silo mentality (fishbowl effect, internal focus on program goals, program concerns are often short-sighted and inward)

Table 47 – Continued: Barriers to Knowledge Integration

Channels	Barrier Instances
3. Function – Program	22. Custom program needs (program resistance against standardization and commonality due to preference for customized tools/templates in place) 23. Databases not efficient (databases not always up-to-date, information is often missing, irrelevant, obsolete, vague, not trusted, difficult to search due to information overflow) 24. Informal problem solving processes (no structured approach to learning, you don't always get invited to the right meeting 25. Lack of formal process (knowledge sharing not well planned, mostly by circumstance not by design, you have to ask others for help but no formal way of identifying who the experts are, informal capture and sharing of lessons learned, no institutionalized "post-mortems") 26. Lack of defined authority (no authority to enforce common tools/processes on programs) 27. Lack of good framework (no business or technology model to determine what knowledge to standardize, or which technology areas should be common across programs, which knowledge to share with whom, how to move the right person for the right need and best fit) 28. Limited financial resources (limited training budgets) 29. Limited human resources (not enough subject matter experts on system architectures, only one person to oversee knowledge management in the entire organization) 30. Mediation limitations (costly for functions to bridge divide between programs, not true integration, some tension on function trying to serve all programs) 31. Organizational distance (programs seek help from function only as a last resort, functional groups not always party to program design work and reviews, programs often unaware of all the resources available to them from functional groups) 32. Organizational structure limitations (tendency of functional structure to keep specialists connected with their knowledge base, but at the cost of weak knowledge integration across different domains) 33. Tool limitations (e.g. tools too complex, too numerous, deployed late or with bugs; new tools have limited use due to incompatibility with legacy tools, and standardized tools are so generic that they're not useful; most tools not flexible enough for complex problem solving) 34. Training limitations (e.g. ineffective formal training, repetitive and too general instead of sharpening skills and learning from past problems; formal mentoring is spotty due to forced pairings that don't work very well)
4. Prime – Supplier	35. Arm's length relationship (purchase order mentality, contractor-sub relationship, suppliers forgotten about in coordination process, supplier not notified of design changes) 36. Bureaucratic restrictions (restricted access to facilities, delays from indirect/mediated knowledge sharing) 37. Classified information (no foreign sharing of technical information; clearance delays) 38. Concurrent engineering limitations (concurrent tools don't always reflect the latest changes, system interdependencies hidden, supplier doesn't know impact of his subsystem problem on overall system, hinders accurate sharing of problem information) 39. Conservative mentality (when in doubt err conservatively and don't share) 40. Contractual restrictions (fixed priced contracts makes suppliers risk averse and unwilling to share immature innovations since potential problems mean they have to swallow the cost; disorganized sharing in contractual letters and direct correspondence between engineers)

Table 47 – Continued: Barriers to Knowledge Integration

Channels	Barrier Instances
4. Prime – Supplier	41. Cultural barriers (organizational rivalry, "not invented here" syndrome, belief that internal decisions/tools are the best ones, program biases for or against supplier X, international cultural/language barriers, time zone differences make communication harder) 42. Firefighting culture (multiple parallel developments at prime and supplier, no coordination, a source of requirements not matching) 43. Information variability (variability depending on prime-supplier relationship; performance requirements not as detailed as military specifications, variability depending on experience of prime engineer writing requirements and supplier engineer interpreting them) 44. Lack of formal process (tendency in prime to do business informally, lots of information conveyed by people walking to other people's desks, informal decisions not flowed down, undermines formal sharing in regular meetings and teleconference calls) 45. Limited financial resources (need a lot of money to build and maintain supplier network, costly to switch suppliers which reduces innovation) 46. Network limitations (tensions on team members having different organizational loyalties, e.g. supplier representative works on prime IPT but reports to supplier, difficult to bring in new suppliers into the network since it can break established relationships) 47. Political restrictions (some forced selection of suppliers for political considerations even if not highest value for knowledge sharing, such as international partnering) 48. Proprietary information (no sharing of engineering and manufacturing discoveries, financial and cost information, patented knowledge) 49. Tool limitations (intranets do not include suppliers, sharing databases with supplier limited to administrative information) 50. Trust issues (supplier doesn't reciprocate open sharing, late notification to prime of strategic issues due to competition, lack of trust by prime and desire to control suppliers so relationship still based on specifications instead of flowing down customer requirements)
5. Corporate – Program	51. Limited learning opportunities (one plane every 15 years, limited and slow learning; few new aircraft programs, no new talent development) 52. Poor knowledge coordination (excessive focus on training and hiring to fulfill needs, but poor planning and matching, job rotation does not take people dynamics in consideration)
6. Customer – Prime	53. Indirect channels (have to go through customer to get clearance, causes delays and limits sharing; need customer approval to change a spec, takes a long time, barrier to problem solving efficiency)
7. Industry – Prime	54. Limited knowledge resources (lifecycle knowledge is getting scarce across the industry) 55. Obsolescence issues (outdated information in industry communities of practice)
8. Supplier – Supplier	56. Communication barriers (can't talk to other subsystem IPT's across companies)

Table 48: Most Frequently Cited Concepts and Properties in the Interview Data

Concepts / Properties	Frequency (No. of Times Cited)	% of Total
Requirements	123	0.3
Architecture	121	0.29
Lessons	85	0.21
Component	41	0.1
Reviews	39	0.1
Communication	35	0.09
Experience	35	0.09
Meetings	34	0.08
Training	34	0.08
Commonality	31	0.08
Testing	30	0.07
Program-to-program	28	0.07
Formal	24	0.06
Informal	24	0.06
Proprietary	23	0.06
Contract	21	0.05
Database	21	0.05
Intra-program	20	0.05
Expertise	19	0.05
Prime-to-supplier	18	0.04
Assistance	15	0.04
Coordination	15	0.04
Troubleshooting	13	0.03
Co-location	13	0.03
Experts	12	0.03
Advice	12	0.03
Mentoring	11	0.03
Collaboration	11	0.03
Function-to-program	11	0.03
Concurrent	10	0.02
Long-term (for comparison with above)	1	0

APPENDIX C: SUMMARY TABLES FROM QUESTIONNAIRE DATA REDUCTION

The tables in this appendix provide a summary of the problem solving and knowledge integration data collected with the structured questionnaire in the final round of interviews across all three case studies.

Table 49: Problem and System Data

No.	Problem Type	Affected System	System Architecture	Tech. Maturity	Problem Novelty	Problem Severity
1	S/W integration	Mission Computer	Integral	Old	Low	Mission
2	H/W integration	Bomb Rack	Modular	Old	High	Safety
3	H/W design	MF Radar	Modular	Old	Low	Mission
4	System design	Radar Altimeter	Modular	New	High	Safety
5	Sensor fusion	Mission Computer	Integral	Old	High	Mission
6	Sensor fusion	Electronic Warfare	Integral	Old	High	Mission
7	H/W design	Electronic Warfare	Integral	Old	Low	Mission
8	S/W integration	FLIR System	Modular	New	High	Mission
9	System integration	Autopilot	Modular	Old	High	Mission
10	H/W design and integration	Display	Integral	New	Low	Low
11	H/W design and integration	Fwd Radar	Modular	New	High	Mission
12	S/W design	GPS Guidance	Modular	Old	Low	Mission
13	System integration	Display	Integral	New	Low	Mission
14	H/W design and integration	AIFF	Modular	New	High	Mission
15	H/W design	MF Radar	Modular	New	Low	Mission
16	System integration	Mission Computer	Integral	New	Low	Mission
17	H/W design	R/F Cabling	Modular	New	Low	Low
18	Sensor fusion	Brake Sensor	Modular	Old	High	Safety
19	H/W design	Navigation	Modular	New	High	Mission
20	S/W design	Communication	Modular	New	Low	Mission
21	System integration	Mission Computer	Integral	New	High	Mission
22	System integration	MF Radar	Modular	New	Low	Mission

Table 49 – Continued: Problem and System Data

No.	Problem Type	System	System Architecture	Tech. Maturity	Problem Novelty	Problem Severity
23	System integration	Mission Computer	Integral	New	Low	Mission
24	System integration	Mission Computer	Integral	Old	Low	Mission
25	System integration	MF Radar	Modular	New	High	Mission
26	H/W design	Canopy	Modular	New	Low	Safety
27	H/W design	Electronic Warfare	Integral	New	Low	Mission
28	S/W integration	Mission Computer	Integral	New	High	Mission
29	S/W integration	Mission Computer	Integral	New	High	Mission
30	H/W integration	MF Radar	Integral	New	High	Mission
31	S/W integration	Indicator Instrument	Modular	Old	High	Mission
32	System integration	Display	Modular	New	High	Mission
33	S/W integration	Network	Integral	Old	Low	Low
34	Sensor fusion	Navigation	Integral	Old	High	Mission
35	H/W design	Mission Computer	Integral	Old	Low	Mission
36	S/W integration	Electronic Warfare	Integral	New	High	Mission
37	S/W design	Electronic Warfare	Integral	New	High	Mission
38	H/W design	Flight Actuators	Integral	New	High	Safety
39	H/W integration	Flight Controls	Integral	New	High	Safety
40	System integration	Mission Computer	Integral	Old	High	Safety
41	System design	Communication	Integral	New	Low	Mission
42	S/W design	Data Module	Modular	New	High	Mission
43	Sensor fusion	Navigation	Integral	New	Low	Mission
44	H/W integration	MF Radar	Modular	New	Low	Mission
45	H/W design	Circuit Board	Modular	New	High	Low
46	System integration	Electronic Warfare	Integral	New	High	Mission
47	H/W design	Memory Device	Modular	Old	High	Low
48	System integration	Navigation	Integral	Old	High	Mission
49	H/W design	Mission Computer	Integral	New	Low	Mission

Table 50: Problem Solving and Knowledge Integration Data

No.	Problem Spread (Other Affected Systems)	Internal Boundaries Crossed	External Boundaries Crossed	Information Exchanged	Advice Exchanged	Assistance Exchanged
1	4	1	1	x	x	
2	5	2	2	x		x
3	5	2	1	x		x
4	1	0	3	x	x	x
5	4	1	1	x	x	x
6	5	1	1	x		x
7	5	2	1	x	x	
8	2	1	2	x		x
9	2	2	0	x		x
10	3	2	1	x	x	x
11	2	2	2	x	x	x
12	1	0	1	x		
13	3	1	1	x		x
14	2	1	2	x		x
15	1	1	1	x		x
16	2	0	2	x	x	x
17	1	0	1		x	x
18	3	1	1	x		x
19	1	0	3	x	x	x
20	1	0	1	x		x
21	3	2	3	x	x	x
22	5	2	1	x		x
23	5	1	2	x		x
24	1	0	2	x	x	x
25	1	0	3	x		x
26	1	1	1	x	x	x
27	1	0	2	x	x	x
28	2	1	1	x		x

Table 50: Problem Solving and Knowledge Integration Data

No.	Problem Spread (Other Affected Systems)	Internal Boundaries Crossed	External Boundaries Crossed	Information Exchanged	Advice Exchanged	Assistance Exchanged
29	2	1	2	x	x	
30	2	1	1	x	x	x
31	1	0	3	x	x	x
32	2	1	2	x		x
33	5	1	1	x	x	
34	2	1	2	x		x
35	1	2	1	x		x
36	2	2	2	x		x
37	1	1	2	x	x	x
38	5	2	2	x		x
39	5	2	1	x		x
40	5	2	1	x		x
41	5	2	1	x	x	
42	1	1	1	x	x	x
43	2	2	0	x	x	x
44	2	0	1	x		x
45	1	0	2	x	x	x
46	5	2	2	x	x	x
47	1	0	3	x	x	x
48	3	1	1	x		x
49	5	2	2	x	x	x

Table 51: Knowledge Integration Data

No.	Mechanisms on Channel #1	Mechanisms on Channel #2	Mechanisms on Channel #3	Mechanisms on Channel #4	Mechanisms on Channel #5
1		Problem reports, telecons			Test data
2	Chief engineer, requirements, specifications	Integrated design tools, engineering change docs, simulation		Graybeards, design review	Site visits, requirements, specifications, engineering change docs
3	Lessons learned database, requirements, specifications, wizards	Integrated design tools, engineering change docs, status meetings			Status meetings, requirements, specification, test data, engineering change docs
4			Share sessions, tiger team, subject matter experts	Tech fellows, tiger team	Site visits, tiger team, engineering change docs, requirements, specifications
5		Problem reports, s/w change docs, troubleshooting meetings		Tiger team	
6		Problem reports, s/w change docs, troubleshooting meetings		Tech fellows	
7	Gurus, requirements, specifications	troubleshooting meetings			Requirements, specifications, engineering change docs, telecons
8	System IPT lead, lessons learned database, wizards			Tiger team, Subject matter experts, tech fellows	Test data, tiger team
9	Tech fellow, troubleshooting meetings, requirements, specifications	Subject matter experts, troubleshooting meetings, simulation			
10	Guru	Interface control docs, simulation			Status meetings, site visits, engineering change docs

Table 51 – Continued: Knowledge Integration Data

No.	Mechanisms on Channel #1	Mechanisms on Channel #2	Mechanisms on Channel #3	Mechanisms on Channel #4	Mechanisms on Channel #5
11	System IPT lead	Interface control docs	Design review		Requirements, specifications, telecons, site visits, engineering change docs
12					Test data, s/w change docs
13		Problem reports, interface control docs, s/w change docs, simulation, wizards			Interface control docs, test data, engineering change docs
14	Chief engineer			Subject matter experts	Co-location, requirements, specifications, engineering change docs, tech fellows, subject matter experts
15	Chief engineer, graybeard, status meetings, requirements, specifications				Requirements, specifications, engineering change docs, Co-location
16				Tiger team	Site visits, test data, tiger team, telecons, engineering change docs,
17				Design reviews, material integrator, liaison engineer	
18		Troubleshooting meeting, interface control docs, problem reports		Tech fellows	
19			Share sessions, taskforce	Taskforce	Taskforce, site visits, test data
20					Test data, site visits, problem reports, s/w change docs simulation

Table 51 – Continued: Knowledge Integration Data

No.	Mechanisms on Channel #1	Mechanisms on Channel #2	Mechanisms on Channel #3	Mechanisms on Channel #4	Mechanisms on Channel #5
21	Chief engineer	Taskforce, s/w change docs, troubleshooting meetings, problem reports,	Taskforce	Taskforce	Taskforce, test data
22	Subject matter expert	Troubleshooting meetings, interface control docs, problem reports			Test data, engineering change docs
23		Problem reports, troubleshooting meetings, wizards		Design review	Subject matter experts, interface control docs, test data, engineering change docs, site visits
24			Subject matter experts, tiger team	Subject matter experts, tiger team, lessons learned docs	
25			Subject matter experts, tiger team	Tiger team, subject matter experts, tech fellow	Test data, tiger team, requirements, specifications
26	Chief engineer, requirements, specifications				Requirements, specifications, engineering change docs, status meetings, problem reports
27			Design review, subject matter expert		Requirements, specifications, status meetings, site visits, engineering change docs
28		Troubleshooting meetings, problem reports, interface control docs, databases			Requirements, specifications, engineering change docs, status meetings, site visits
29		Problem reports, troubleshooting meetings		Red team	Engineering change docs

Table 51 – Continued: Knowledge Integration Data

No.	Mechanisms on Channel #1	Mechanisms on Channel #2	Mechanisms on Channel #3	Mechanisms on Channel #4	Mechanisms on Channel #5
30		Requirements, specifications, troubleshooting meetings, engineering change docs		Subject matter experts	
31			Lessons learned docs, red team	Tech fellows, red team	Interface control docs
32		Problem reports, interface control docs		Tech fellows, subject matter experts	Interface control docs
33		Interface control docs, databases, integrated design tools		Design review, graybeards	
34		Interface control docs, s/w problem reports		Tech fellows	Telecons
35	Wizards	Interface control docs, s/w problem reports	Subject matter experts		
36	Wizards, tiger team, requirements, specifications	Tiger team, interface control docs, s/w problem reports, s/w change docs		Tiger team	Tiger team, requirements, specifications, test data
37	System architect, requirements, specifications			Tiger teams, tech fellows	Site visits, co-location
38	Tiger team, requirements, specifications	Tiger team, problem reports, troubleshooting meetings		Tiger team	Tiger team, site visits, requirements, specifications, engineering change docs
39	Tiger team, requirements, specifications	Tiger team, interface control docs, problem reports, troubleshooting meetings		Tiger team	
40	System architect, requirements, specifications	Tiger team, interface control docs, integrated design tools			Tiger team, engineering change docs

Table 51 – Continued: Knowledge Integration Data

No.	Mechanisms on Channel #1	Mechanisms on Channel #2	Mechanisms on Channel #3	Mechanisms on Channel #4	Mechanisms on Channel #5
41	System IPT leads, Requirements, specifications, design review	Interface control docs, integrated design tools, engineering change docs			Requirements, specifications, engineering change docs
42	Chief engineer, System architect				Requirements, specifications, site visits, engineering change docs
43	Liaison engineer	Interface control docs, problem reports, s/w change docs, troubleshooting meetings			
44					Requirements, specifications troubleshooting meetings, engineering change docs
45			Design review		Requirements, specs, site visits, engineering change docs
46	Chief engineer, Lessons learned docs, subject matter experts	Interface control docs, problem reports, troubleshooting meetings, subject matter experts		Subject matter experts	Requirements, specifications, site visits, subject matter experts engineering change docs
47			Share sessions, design reviews	Subject matter experts, tiger team	Requirements, specifications, engineering change docs, tiger team
48		Problem reports, troubleshooting meetings, s/w change docs			Requirements, specifications, engineering change docs
49	IPT leads, decision board review	Interface docs, problem reports, troubleshooting meetings		Subject matter experts	Requirements, specifications, engineering change docs

APPENDIX D: STATISTICAL CALCULATIONS

This appendix illustrates the statistical calculation of the chi-square statistic for testing whether the use of a particular knowledge integration channel is sensitive to differences in system architectures or if the observed differences are random. The following shows a smaller calculation which excludes channels #1 and #4 since they are used as a "last recourse" for knowledge integration and therefore are independent from the other three channels shown here.

Observed Values: Each number in the table below represents the observed instances when a particular channel was used to integrate knowledge during problem solving in a particular architecture regime

	horizontal count	1	1	1	3
vertical count	**System Architecture**	**Ch. #2**	**Ch. #3**	**Ch. #5**	*Totals*
1	**Integral**	22	4	17	43
1	**Modular**	8	8	19	35
2	*Totals*	30	12	36	78

Expected Values: Calculated values of expected frequencies Eij per the guidelines of the chi-square test, where Eij = (ith row total) x (jth column total) / (grand total)

	Ej1	**Ej2**	**Ej3**
E1i	16.54	6.62	19.85
E2i	13.46	5.38	16.15

Chi-Square Calculations: Calculated chi-square values $\chi^2ij = (Oij - Eij)^2 / Eij$

where O = observed value, E = expected value

O	**E**	**|O-E|**	**(O-E)^2**	**(O-E)^2/E**
22.00	16.54	5.46	29.83	1.80
4.00	6.62	2.62	6.84	1.03
17.00	19.85	2.85	8.10	0.41
8.00	13.46	5.46	29.83	2.22
8.00	5.38	2.62	6.84	1.27
19.00	16.15	2.85	8.10	0.50

Chi-Square Statistic and Degrees of Freedom (DF): Calculated chi-square χ^2 = sum of all χ^2ij where degrees of freedom DF = (horizontal count – 1) * (vertical count – 1)

χ^2 (chi-square value):	7.23
Degrees of Freedom (DF):	2

Chi-Square Distribution Table: Values obtained from the chi-square curve for given DF and alpha levels of significance where alpha = 0.05 correlates with 95% significance level

DF	0.1	0.05	0.025	0.01	0.005
1	2.71	3.84	5.02	6.63	7.88
2	4.61	5.99	7.38	9.21	10.60
3	6.25	7.81	9.35	11.34	12.84
4	7.78	9.49	11.14	13.28	14.86
5	9.24	11.07	12.83	15.09	16.75
6	10.64	12.59	14.45	16.81	18.55
7	12.02	14.07	16.01	18.48	20.28
8	13.36	15.51	17.53	20.09	21.95
9	14.68	16.92	19.02	21.67	23.59
10	15.99	18.31	20.48	23.21	25.19

$\chi^2 = 7.23 > p\ (0.05) = 5.99$ means the observed values and expected values are far apart, therefore it is possible to reject the null hypothesis Ho: there is no relationship between system architecture and knowledge integration (i.e. integral and modular differences are normal error) and to accept the alternative hypothesis H1: there is a relationship between system architecture and knowledge integration (i.e. differences between integral and modular are not coincidence)

Note: The Yates correction factor was not used despite one observed value = 4 < 5 (Yates correction overcompensates and is too conservative)

APPENDIX E: RESEARCH PROTOCOL

All data collection was conducted in accordance with the rules and procedures mandated by the Massachusetts Institute of Technology (MIT) Committee on the Use of Humans as Experimental Subjects (COUHES, **http://web.mit.edu/committees/couhes**). Non-disclosure agreements were signed by the author with the participating organizations and additional provisions were taken to ensure confidentiality and proprietary protection under the provisions of the research agreement between the Lean Aerospace Initiative (LAI) and its consortium members. The rights and responsibilities of interview subjects were detailed in a research guide prepared by the author and disseminated in advance, as shown below.

RESEARCH GUIDE FOR RESPONDENTS - THESIS ON KNOWLEDGE INTEGRATION IN LARGE-SCALE DEVELOPMENT OF COMPLEX SYSTEMS

1. Introduction

Thank you for your interest in participating in this research project conducted by Marc Haddad, PhD candidate in Engineering Systems and research assistant at the Lean Aerospace Initiative (LAI) at the Massachusetts Institute of Technology (MIT). LAI is a consortium of government, industry, organized labor, and academia in the aerospace industry interested in researching, developing, implementing, and accelerating enterprise transformation based on lean principles and practices. MIT's role in this partnership is to provide objective and systematic research leading to implementation in practice.

You were selected as a possible participant in this research project because of your professional affiliation and current position in the aerospace industry. You should read the information below, and ask questions about anything you do not understand, before deciding whether or not to participate. This research guide provides an overview of the research project and the types of interview questions to be asked of research participants. The guide also covers issues of confidentiality with respect to all data collected from participating organizations.

2. Research Objectives

This research project is part of an MIT doctoral thesis on knowledge integration in complex systems development, with a focus on military avionics as the research lens. The objective is to develop a conceptual framework for how knowledge is transferred, shared and applied in solving major technical problems during the design and build phases of product development. It is motivated by the vested interest that organizations have in evolving their problem-solving capabilities through leveraging their knowledge resources efficiently and effectively. The main value-proposition of this research is in identifying the mechanisms and practices by which engineering knowledge is integrated across team, program, functional and organizational boundaries in solving major technical problems in a large-scale complex development environment. The research seeks to address the following types of questions:

- Knowledge-sharing practices: What are specific channels, strategies and mechanisms employed by your organization to transfer and share engineering knowledge (e.g. communication and coordination channels and mechanisms) during the design and build phases of product development?

- Problem-solving practices: What are specific channels, strategies/practices and mechanisms employed by your organization to apply engineering knowledge (e.g. cooperation and collaboration channels and mechanisms) in solving major technical problems during the design and build phases of product development?

- Policy barriers and enablers: What are the main policies impeding or enabling knowledge transfer/sharing and problem-solving within your organization, as well as with other partners/suppliers?

3. Potential Benefits

The research is expected to provide the following outcomes: 1) a doctoral thesis with an original contribution to the field of engineering systems in terms of a framework and heuristics for

knowledge integration in large-scale complex systems development, 2) a conference paper providing a high-level summary of key findings for communication to LAI consortium members, 3) presentation of key findings at Lean Aerospace Initiative meetings, workshops and conferences, and 4) publication of one or more papers in scholarly journals. These research products will be made available through the LAI website (**http://lean.mit.edu**)

4. Research Questions

Participants will be asked a series of questions about knowledge integration strategies and mechanisms used in their organization in a series of 1 to 1.5 hour interviews. Interview questions will be focused on the types of practices used to exchange knowledge within the participating organization and with other partner/supplier organizations in the development of complex military avionics systems. Questions will be generic in nature and will not require any classified, sensitive, confidential or proprietary information. The main types of questions are:

a) General questions: Participants will be asked to provide a brief overview of their educational background and professional experience, particularly as these pertain to their current role in solving technical problems related to the development of a particular mission subsystem.

b) Knowledge transfer/sharing questions: Participants will be asked about the knowledge transfer and sharing channels, strategies/practices and tools/mechanisms they use most frequently to exchange knowledge, and will be asked to evaluate their effectiveness in helping them conduct technical problem solving during development.

A survey-like questionnaire will be administered separately to collect data about specific practices used to solve major technical problems in the development of complex avionics systems. The data will be coded to strip any personal identifiers. These questions relate to:

c) Problem-solving questions: Participants will be asked to illustrate specific problem-solving situations with their particular avionics subsystem and how they exchanged and applied knowledge to diagnose and solve the problem. The questionnaire is designed to collect data

about problem description, the characteristics of the avionics subsystem(s) affected by the problem and the core technologies involved, the problem solving approach (or approaches) taken, the stakeholders involved and the channels and mechanisms used to exchange and apply knowledge during problem solving.

Note: see Appendices A.1 and A.2 for samples of actual questions used in the interview process and Appendix A.3 for the questionnaire form used in the problem solving survey.

5. Research Participants

The participants of most interest for this research are: 1) program managers, chief engineers and material managers who are familiar with the "big picture" and major issues related to knowledge transfer/sharing and problem solving in your organization, 2) knowledge management experts in your company or program/division, and 3) IPT leads and engineers directly involved in solving design and integration problems in the development of major mission subsystems (Radar, EW, CNI, Mission Computer) in the aircraft programs under current development.

In the past, case studies at LAI stakeholder member organizations have been greatly facilitated by the appointment of a single individual to serve as the main point-of-contact or interface between the host organization and the MIT research team. Therefore, it is requested that the host organization for the subject case study identify a particular person to serve as the main point of contact for the prospective case study, with responsibility for logistics and coordination in the course of the case study.

6. Protection of Data Confidentiality

The confidentiality of all data provided to MIT Lean Aerospace Initiative (LAI) researchers that are otherwise not available publicly (e.g., business-sensitive plans, practices, processes or technologies) shall be strictly protected in conformance with the data confidentiality and proprietary information provisions of the MIT LAI Consortium Agreement with all industry

sponsors. The same strict proprietary data protection provisions shall be extended to all responding organizations that are not sponsoring members of the Lean Aerospace Initiative.

A nondisclosure agreement between the MIT LAI researchers and an individual organization will be signed should that organization consider such a nondisclosure agreement a necessary condition for its response to this structured case study interview. No information that is provided by an individual or organization, even if it is not considered proprietary or confidential by the individual or organization providing such information, shall be presented or published in a way that would permit the identification of any individual respondent and, moreover, the identify of no specific organization or any of its programs (products) will be identified by name without that organization's prior written permission.

Data will be stored securely at LAI until research is complete, at which time any identifying data will be destroyed. Reported data will be non-attributable. All LAI material is held behind a two lock system. To protect against inadvertent release of information both LAI Research Assistants and Research Staff who are part of this study are provided orientations at the beginning of the academic year and an LAI Orientation Manual addresses these considerations.

7. Rights of Research Subjects

You are not waiving any legal claims, rights or remedies because of your participation in this research study. If you feel you have been treated unfairly, or you have questions regarding your rights as a research subject, you may contact the Chairman of the Committee on the Use of Humans as Experimental Subjects, MIT, Room E23-230, 77 Massachusetts Ave, Cambridge, MA 02139, phone (617) 253-4909.

8. Informed Consent Form

We would like to emphasize that participation in this research is completely voluntary. You are free to refuse to answer any question you are either uncomfortable with or uncertain about. You are also free to withdraw your participation at any time. We understand that you may have

concerns about confidentiality. Several measures will be taken to ensure that your responses will remain confidential. Data will be accessed and processed only by MIT Lean Aerospace Initiative researchers directly engaged in the case study research project. All analysis of the data will be represented in the form of aggregated statistics or interpretations. Excerpts from interviews with specific individuals may be included in the aggregated analysis or research results, but under no circumstances will there be any attribution to your name, your company/program/division name or any identifying characteristics that can be traced to you. Specific practices, processes or technologies that may be considered proprietary or business sensitive will be accorded similar protection of data confidentiality. We understand that the success of any research depends upon the quality of the information on which it is based, and we take seriously our responsibility to ensure that any information you entrust to us will be protected.

Please sign this form to show that you have read and had the opportunity to clarify the contents of this document.

Signature: _____ Date: _____

Do you agree to have excerpts from your interview reported in the form of a quotation (without attribution to you by name)?

❒ Yes ❒ No

Please print legibly (or attach a current business card):

Name: _____

Position: _____

Organization: _____

Phone/Fax: _____

E-mail: _____