



Software Engineering Institute

Results of SEI Independent Research and Development Projects

Len Bass, Dionisio de Niz, Jörgen Hansson, John Hudak, Peter H. Feiler, Don Firesmith, Mark Klein,
Kostas Kontogiannis, Grace A. Lewis, Marin Litoiu, Daniel Plakosh, Stefan Schuster, Lui Sha,
Dennis B. Smith, & Kurt Wallnau

July 2008

TECHNICAL REPORT
CMU/SEI-2008-TR-017
ESC-TR-2008-017

Unlimited distribution subject to the copyright.

<http://www.sei.cmu.edu>



CarnegieMellon

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

Copyright 2008 Carnegie Mellon University.

NO WARRANTY

THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

Internal use. Permission to reproduce this document and to prepare derivative works from this document for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use. Requests for permission to reproduce this document or prepare derivative works of this document for external and commercial use should be directed to permission@sei.cmu.edu.

This work was created in the performance of Federal Government Contract Number FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The Government of the United States has a royalty-free government-purpose license to use, duplicate, or disclose the work, in whole or in part and in any manner, and to have or permit others to do so, for government purposes pursuant to the copyright license under the clause at 252.227-7013.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our website (<http://www.sei.cmu.edu/publications/pubweb.html>).

Table of Contents

Abstract	v
1 Introduction	1
1.1 Purpose of the SEI Independent Research and Development Program	1
1.2 Overview of IRAD Projects	1
2 Improving Architectural Design Through Organizational Considerations	2
Len Bass	2
2.1 Purpose	2
2.2 Background	2
2.3 Approach	4
2.4 Collaborations	4
2.5 Evaluation Criteria	4
2.6 Results	4
2.7 References	7
3 Performance Challenges of Modern Hardware Architectures for Real-Time Systems	8
Dionisio de Niz, Jörgen Hansson, John Hudak, and Peter H. Feiler	8
3.1 Purpose	8
3.2 Background	8
3.3 Approach	11
3.4 Collaborations	13
3.5 Evaluation Criteria	14
3.6 Results	14
3.7 Publications and Presentations	15
3.8 References	15
4 A Research Agenda for Service-Oriented Architecture	17
Grace A. Lewis (Lead), Kostas Kontogiannis, Marin Litoiu, Stefan Schuster, and Dennis B. Smith	17
4.1 Purpose	17
4.2 Background	17
4.3 Approach	18
4.4 Collaborations	18
4.5 Results	18
4.6 Publications and Presentations	25
4.7 References	26
5 A Software System Engineering Approach for Fault Containment	28
Peter H. Feiler, Dionisio de Niz, Jörgen Hansson, Lui Sha, and Don Firesmith	28
5.1 Purpose	28
5.2 Approach	30
5.3 Collaborations	31
5.4 Evaluation Criteria	31
5.5 Results	32
5.6 Publications and Presentations	34
5.7 References	34

6	Using the Vickrey-Clarke-Groves Auction Mechanism for Enhanced Bandwidth Allocation in Tactical Data Networks	37
	Mark Klein, Daniel Plakosh, and Kurt Wallnau	37
6.1	Purpose of this Research	37
6.2	Background	37
6.3	Computational Mechanism Design	38
6.4	Approach	38
6.5	Auctioning Bandwidth Allocation on Tactical Networks	39
6.6	Assessing the Auction	41
6.7	Contribution of this Work	42
6.8	Conclusions	43
6.9	References	44

List of Figures

Figure 2-1: The Relationship Between Module Dependency and Team Coordination	4
Figure 2-2: Factors Impacting Coordination	5
Figure 3-1: Speedup to Cache Allocation Function	13
Figure 3-2: Two Task QRAM Cache Assignment	13
Figure 4-1: Overview of the SOA Problem and Solution Space	19
Figure 4-2: Expanded View of the SOA Problem and Solution Space	20
Figure 4-3: Mapping Between Phases, Activities, and Indicators	21
Figure 4-4: SOA Research Taxonomy	22
Figure 6-1: Tactical Display With Fused Track Data	40
Figure 6-2: Studying the Runtime Effects of the Auction	42

List of Tables

Table 2-1: Topics of Coordination from One Development Project	5
Table 2-2: Styles of Coordination and Type of Tool Used for Each Style	6

Abstract

The Software Engineering Institute (SEI) annually undertakes several independent research and development (IRAD) projects. These projects serve to (1) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) support further exploratory work to determine whether there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IRAD projects that were conducted during fiscal year 2007 (October 2006 through September 2007).

1 Introduction

1.1 PURPOSE OF THE SEI INDEPENDENT RESEARCH AND DEVELOPMENT PROGRAM

SEI independent research and development (IRAD) funds are used in two ways: (1) to support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) to support further exploratory work to determine whether there is sufficient value in eventually funding the feasibility study work as an SEI initiative. It is anticipated that each year there will be three or four feasibility studies and that one or two of these studies will be further funded to lay the foundation for the work possibly becoming an initiative.

Feasibility studies are evaluated against the following criteria:

- Mission criticality: To what extent is there a potentially dramatic increase in maturing and/or transitioning software engineering practices if work on the proposed topic yields positive results? What will the impact be on the Department of Defense (DoD)?
- Sufficiency of study results: To what extent will information developed by the study help in deciding whether further work is worth funding?
- New directions: To what extent does the work set new directions as contrasted with building on current work? Ideally, the SEI seeks a mix of studies that build on current work and studies that set new directions.

1.2 OVERVIEW OF IRAD PROJECTS

The following research projects were undertaken in FY 2007:

- Improving Architectural Design Through Organizational Considerations (Len Bass)
- Performance Challenges of Modern Hardware Architectures for Real-Time Systems (Dionisio de Niz, Jörgen Hansson, John Hudak, and Peter H. Feiler)
- A Research Agenda for Service-Oriented Architecture (Grace A. Lewis [Lead], Kostas Kontogiannis, Marin Litoiu, Stefan Schuster, and Dennis B. Smith)
- A Software System Engineering Approach for Fault Containment (Peter H. Feiler, Jörgen Hansson, Dionisio de Niz, Lui Sha, and Don Firesmith)
- Using the Vickrey-Clarke-Groves Auction Mechanism for Enhanced Bandwidth Allocation in Tactical Data Networks (Mark Klein, Daniel Plakosh, and Kurt Wallnau)

These projects are summarized in this technical report.

2 Improving Architectural Design Through Organizational Considerations

Len Bass

2.1 PURPOSE

Architectural design decisions and organizational coordination mechanisms must be aligned in order for development to proceed smoothly. Our goal is to make the factors of this alignment precise and to use this understanding to provide guidance to designers and managers. The guidance for designers will be of the form: “in this type of organizational environment with this type of organization coordination mechanisms, these are the decisions that may be problematic.” The guidance for managers will be of the form: “these are the types of coordination mechanisms that should be in place to support these particular architectural decisions.” We also expect to develop indicators that managers can use to provide early indications of potential organizational or architectural misalignment.

2.2 BACKGROUND

It is well known that architectures for both software and systems become deeply “embedded” in the organizations that design and build software-intensive systems (e.g., [Conway 1968]). Architectures determine key characteristics of organizations, such as work assignments, and have implications for communication patterns, habitual ways that people select and filter information, and organizational problem-solving strategies. Changing the architecture, even in seemingly simple ways, can cause serious misalignments between the architecture and the organization, leading in some cases to complete failure of the firm [Henderson 1990]. Organizations are notoriously difficult to change, and we do not yet have any tools for understanding the kinds of changes we are imposing on them when we perform architectural design. We don’t know how to assess the risks, nor do we know how to address the risks once they are identified.

Suppose, for example, that the designer of an architecture is considering two alternative designs. The designer knows that design one requires the developers of a particular component to access expertise from a separate organization, requiring extensive long-distance communication. Design two requires the same developers to access expertise from a co-located site. Everything else being equal, design two is preferable because it will reduce organizational risk by lowering the communication overhead required to solve any problem that arises. Organization structure, existing communication patterns, location of people and resources, shared work history, and potentially other factors not yet identified will influence the risk, time required, component quality, and development efficiency of the design and construction of software components. Organizational factors should be considered as design decisions are made, just as performance or reliability factors are considered.

It is also well known that a system’s software architecture is a central artifact in the development of software-intensive systems. It is a primary determinant of the system’s quality attributes. It is the bridge between business goals served by a system and that system’s implementation. The SEI

has invested considerable effort in understanding the principles of design and analysis of software architecture. However, people must effectively collaborate to realize the benefits of software architecture. Conversely, organizational inhibitors to collaboration can be inimical to realizing the potential benefits of software architecture. Therefore understanding the relationship between types of organizations and types of architectures is vital to realizing the benefits of software architecture.

The purpose of this project was to investigate the relationship between organizational characteristics and software architecture. If two people work together on the same component, they must communicate about all aspects of the component. If they work on different components, they must communicate about the ways in which their respective components interact. The two situations may require different types and bandwidths of communication. Depending on the fashion in which two components interact, there may be a requirement for high or low bandwidth communication. The communication bandwidth among people is affected by many different elements such as co-location, organizational structure, processes used, technological support, and knowledge of where different types of expertise reside in an organization. In the modern world, most large systems are constructed across multiple sites, usually involving multiple distinct organizations. Understanding the organizational characteristics that are essential to effectively building a system with a particular architecture will provide fundamental knowledge that can be exploited to more effectively manage and design large software systems.

If the relationship between organizational characteristics and software architecture were well understood, it would have application in architecture evaluation and architecture design, and in assessing the readiness of an organization for architecture-centric development. During an architecture evaluation, discovering a misalignment between architectural decisions and organizational structure amounts to discovering a risk to the development effort. During an architecture design, discovering a misalignment between proposed architectural decisions and proposed organizational structure allows for the correction of the misalignment or at least for a clearly understood allowance of the misalignment in the project management. There might be a tradeoff, for example, between optimizing the performance of a system and optimizing it for the organizational structure of the developing organizations. One question the SEI Software Architecture Technology initiative frequently hears is, “How ready is organization X for architecture-based development?” Being able to analyze the structure of organization X with respect to common architecture usages is one aspect of being able to answer this question. A fundamental understanding of the relationship between organizational characteristics and software architectures could allow us to go beyond assessing readiness in general, and determine the range of architectures an existing organization could build.

A complicating element of the understanding of organizational characteristics is that they evolve over time, and desirable characteristics may change, depending on the current stage of project development. During the initial stages of a project, one might see, for example, particular patterns of communication among the groups working on the infrastructure. During later stages one might see communication among application developers and infrastructure developers.

During this research project, we have examined the factors that impact coordination among distributed teams. We have discovered that, in one project, over 50% of the coordination topics were

architectural in nature. We have discovered examples where organizational factors impact coordination about architectural decisions and examples where process factors impact coordination. We have also analyzed the types of tools that are used to perform the coordination in different circumstances.

2.3 APPROACH

The approach we used in this project was empirical. We analyzed data gathered from several development projects. The data included coordination data as well as data gathered from normal development activities, such as configuration management logs.

2.4 COLLABORATIONS

The SEI participant in this study was Len Bass. Collaborators included Prof. James Herbsleb of the School of Computer Science at Carnegie Mellon and one of his graduate students. An additional collaborator was Matthew Bass of Siemens Corporate Research. Matthew Bass's participation was funded by Siemens.

2.5 EVALUATION CRITERIA

Some results of this investigation were peer reviewed and accepted for presentation at an international conference [Cataldo 2007b]. Other results are being written and will be submitted for peer review.

2.6 RESULTS

We investigated the portion of the life cycle after an initial architecture has been defined. This includes development and post release maintenance. Figure 2-1 shows the situation we investigated: Distinct development teams are working on modules that have dependencies between them. These dependencies generate a need for coordination among the development teams over the portions of their respective modules that have dependencies with other modules.

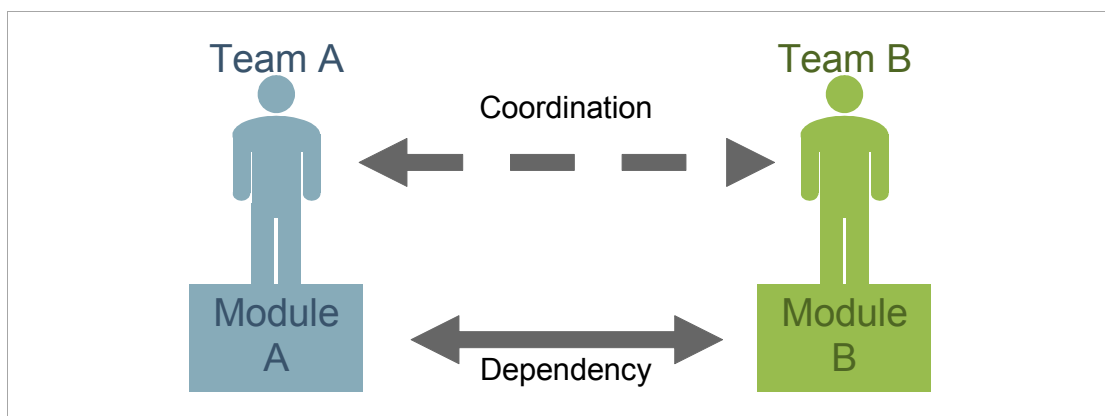


Figure 2-1: The Relationship Between Module Dependency and Team Coordination

One question that occurs in this situation is: what do the teams coordinate about? Table 2-1 shows that more than 50% of the threads of coordination concern architecture or design issues leading to an architectural decision.

Table 2-1: Topics of Coordination from One Development Project

Task	Number of Threads	Percentage
Requirements specification	6	3.8%
overall architecture	30	18.9%
project management	6	3.8%
integration test	1	0.6%
process definition and management	14	8.8%
central infrastructure	11	6.9%
detailed design involving multiple teams—i.e., about architectural issues	58	36.5%
specification of unit tests	9	5.7%
coding	14	8.8%
unit testing	4	2.5%
maintenance of related artifacts	6	3.8%

Figure 2-2 shows the coordination being impacted by several important factors. These factors include the organizational structure and the processes being followed. The coordination occurs utilizing one or more different media types.

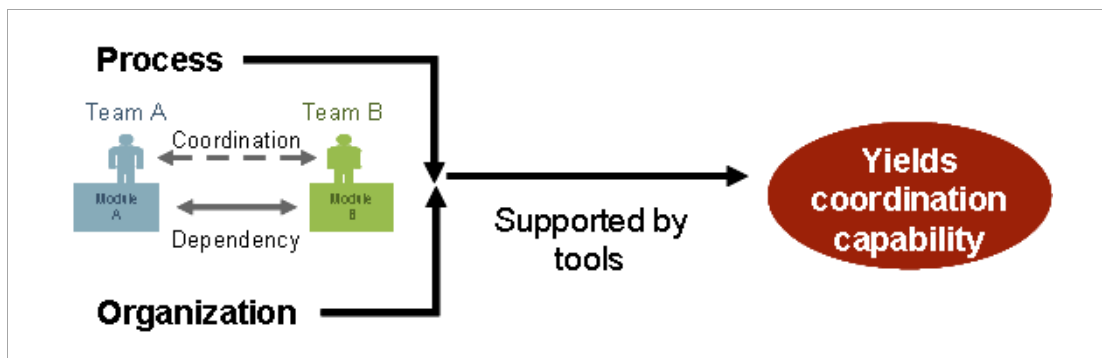


Figure 2-2: Factors Impacting Coordination

During the course of the investigation we identified different incidents that resulted in significant coordination and significant development delay. Several of these incidents could be attributed to the division of responsibilities among different organizational entities and to the schedule as it impacted these different entities. Several of these incidents could be attributed to processes being used for the development, again as it impacted the different teams involved in the incidents.

Table 2-2 shows the styles of coordination and the type of tool used for each style within the projects that were monitored. The percentages in particular cases may add up to greater than 100% because threads may initiate use of one media and be continued using another.

Table 2-2: Styles of Coordination and Type of Tool Used for Each Style

	Teleconference		E-mail		Discussion Forum	
	Occurrence		Occurrence		Occurrence	
Topic Category	#	%	#	%	#	%
Information seek	0	0	0	0	7	7
Information post	1	1	709	39.6	22	21
Information seek-reply	31	28	299	16.7	41	39
Task negotiate	112	100	146	8.2	8	8
Task seek action	10	9	209	11.7	6	6
Task seek decision	0	0	0	0	3	3
Task notify status	112	100	278	15.5	0	0
Task notify decision	0	0	14	0.8	1	1
Problem-solution exchange	5	4	137	7.7	11	10
Problem-solution negotiate	5	4	7	0.4	2	2
Design negotiate	19	17	38	2.1	10	9
Design notify decision	3	3	30	1.7	6	6
Design seek decision	1	1	0	0	9	8

Observe that negotiation is dominated by both e-mail and teleconference yet the action portion of the negotiation—as opposed to questions about status or decisions—more frequently uses e-mail than either teleconference or discussion forums.

2.6.1 Additional Work

These results indicate that architectural decisions are a major topic of coordination and that factors of process and organizational structure as mediated by coordination tools help determine the requirement for coordination and the amount of coordination necessary in particular cases.

In related work, the collaborators on this project examined predictors that could be useful for predicting the necessity of coordination. This related work determined that examining the files that are changed together during a development or maintenance activity provide a good predictor for the amount of coordination that occurs [Cataldo 2007a]. Files that change together is a measure that is not available until development is well under way. Future work is focusing on determining a predictor that is available from the architecture rather than from development artifacts.

2.7 REFERENCES

[Cataldo 2007a]

M. Cataldo. “Dependencies in Geographically Distributed Software Development: Overcoming the Limits of Modularity.” PhD diss., Carnegie Mellon University, 2007.

[Cataldo 2007b]

M. Cataldo, M. Bass, J. Herbsleb, & L. Bass. “On Coordination Mechanisms in Global Software Development.” *Proceedings of the 2007 International Conference on Global Software Engineering*, IEEE Press, 2007.

[Conway 1968]

M. E. Conway. “How Do Committees Invent?” *Datamation* 14, 4 (1968): 28-31.

[Henderson 1990]

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

3 Performance Challenges of Modern Hardware Architectures for Real-Time Systems

Dionisio de Niz, Jörgen Hansson, John Hudak, and Peter H. Feiler

3.1 PURPOSE

Real-time and embedded systems increasingly deploy more advanced, albeit standard, hardware CPU architectures, which provide significant performance enhancements in running tasks compared with past CPU architectures. Among the most important advances are the speed of CPUs and the use of multiple levels of cache to compensate for the slower main memory technology. However, using caching technology in embedded systems has always been problematic. The main problem arises from unpredictable execution time, because of the dynamic nature of caches. Overly conservative estimates of resource usage result in an underutilized system and overly optimistic estimates result in decreased performance and can ultimately cause the system to be overloaded and, thus, fail to satisfy temporal constraints. The advancement of CPU caching and pipelining render existing path-based techniques for determining CPU resource usage inappropriate in practice given the difference in actual resource usage as opposed to derived worst-case estimates assuming non-pipelined and non-cached CPU architectures. The state of the cache, and content of the pipeline due to preemptions, affect the actual execution time of a task. Thus, techniques for benchmarking a task running in isolation cannot be adopted as these are most often misleading, or provide unsatisfactory accuracy and confidence. In other words, because we are not able to predict whether or not a memory word will be in cache, the execution of the instruction that uses this word is equally unpredictable. For this reason a common practice in the development of embedded systems is to disable the cache to make sure we know where the memory words are and how long it would take to access them.

Researchers on multiple projects have been working on the predictability of memory access time using caches and other types of on-chip memory, such as scratchpad memory. These projects ignore two important characteristics of embedded systems: the simplicity of the control flow and the periodic nature of real-time tasks. In this paper we present a scheme to take advantage of both factors. We use memory traces to profile the memory access behavior of tasks, relying on the fact that most control and real-time tasks have small variations in the control flow. We also take into account the periodicity of the tasks to evaluate the total speed up of the system when allocating cache. We then combine these two aspects in a near-optimal static cache allocation to obtain most of the benefit of the cache without introducing unpredictable memory access behavior.

3.2 BACKGROUND

Cache allocation algorithms present in modern processors have two basic deficiencies when used in real-time systems: first, they optimize the average memory access time, disregarding the worst-case access time; and, second, they focus on a single executing thread.

Existing cache memory was designed to optimize the execution of general-purpose programs. Improvements in execution time in this type of programs have focused on the average case, given

that it has a good impact on user perception. In addition, these optimization techniques do not rely on any program information beyond the profiling of its immediate memory access patterns. For real-time systems, the main concern related to execution time is guaranteeing that such an execution ends before a well-defined deadline. In this case a real-time program is a program that executes periodically with a fixed period in general, and with a fairly stable memory access pattern due to the narrow focus of its functionality. This functionality is related to a concrete interaction with its environment, such as avoiding the skidding of the wheels in an ABS system or the inflating of an airbag when a crash is sensed. To verify that such programs finish their periodic execution before their deadlines, priorities are assigned based on the generalized rate-monotonic theory [Sha 1994]. This theory uses the worst-case execution time along with the period to verify that a program will always finish before its deadline. The fact that this theory uses the worst-case execution time makes the design of the cache behavior ill-fitted for real-time systems. This is because caching algorithms will load data into cache based on what was accessed in the past. More frequently than not, the loaded data will access the execution of the following instructions, producing good reductions in memory access time. However, with some small frequency this strategy causes large access delays because the data loaded is not the correct one and hence, the right one needs to be loaded. This sporadic high-cost access is dominated by the frequent gains in the average memory access time. However, for the worst-case access time, in fact, the worst of the high cost access times dominate. As a result, the caching strategy produces the opposite result for the worst-case execution time than the one achieved for the average case. Furthermore, determining the worst-case execution time becomes more difficult because producing the worst-case memory access time is harder with regular caching algorithms. This is one of the main reasons embedded system designers, in general, disable the cache of the processors they use to gain more predictable execution time.

The second problem with general-purpose caching algorithms is that they are designed with a single thread in mind. This implies that they do not consider that the past memory access behavior can change completely when a new thread is brought into the processor. The presence of multiple threads in the processor introduces new and larger cache misses.

A core concern in real-time systems is determining the worst-case execution time (WCET) of a program in order to verify if enough CPU cycles are available to complete periodic execution before a deadline. Deriving the WCET of a task is done by analyzing its execution paths and computing the execution time of each path based on the summation of execution times of the instructions involved in the path. The WCET of the task is thus its longest execution, assuming that a task is not utilizing caching and interleaving, which would reduce the actual execution time significantly, and thus enhance the performance. (Note that the cache hit ratio is normally 0.8-0.96.) However, the use of cache modifies the execution time of the various paths differently depending on the memory access pattern of the current execution that defines what memory is in cache at that moment. This effect can make the execution time reduction due to cache difficult to predict, affecting the predictability of the WCET.

There are several ways to increase the predictability of caches, such as controlling preemption, partitioning the cache, and locking objects to the cache. Controlling preemption by the use of preemption points reduces the number of possible task execution interleavings. It also allows for

analyzing the data locality and access patterns in the non-preemptive sections of the code, more accurately predicting the performance advantage of caching, and using a specific replacement strategy. This is based on the observation that when a task is preempted, its cache behavior in general changes as cache entries loaded by the preempted task are replaced by the preempting task.

Locking selected memory objects to the cache eliminates any uncertainty of the memory access cost, but requires that the cost of preloading the cache with the selected objects be included in the resource analysis. Locking the objects ensures access via the cache, and removes the effect of the replacement strategy for these objects. Regular objects still subject for replacement may be affected as there might be a more significant turnover of objects as the effective space available for regular objects is reduced by the amount of space required for the local objects. The optimization game that one plays is to minimize the number of total replacements for regular objects and, thus, maximize predictability for safety-critical tasks. This is controlled by the static/dynamic number of objects that we lock, a determination of which objects to lock to the cache, and the length of the locking time.

By partitioning a cache for different application tasks, we can isolate tasks and reduce the impact of other tasks in systems, as each partition operates independently. This has the advantage that previously developed techniques for determining the cache effects of running tasks in isolation can be adopted, but this time assuming a task only has a fraction of the cache capacity. Efficient partitioning involves determining the size of the partitions and mapping tasks to the various partitions. The cost of isolating tasks by partitioning includes a reduction in effective cache size, most likely causing an overall decreasing hit ratio from a task perspective, and an increasing amount of time spent across partitions for selecting replacement candidates.

In embedded systems another type of memory as fast as cache has been gaining popularity, i.e., scratchpad memory. This type of memory is directly mapped into the memory address space. It is the responsibility of the programmer/compiler to localize in such memory the objects that would be accessed most frequently. This access scheme contrasts with cache, where the hardware is in charge of loading the most frequently accessed objects into the cache and redirecting their access to it. However, this automatic loading scheme is difficult to predict at design time and, hence, makes the execution time of a program difficult to predict as well. Given the importance of predictability of execution time in real-time systems the unpredictability of caches is a liability that leads the designers to disable it in numerous occasions.

As a result of the unpredictability of cache memory, scratchpad memory is the most common form of fast memory (SRAM) in embedded CPUs today [Brash 2002, Adieletta 2002, Motorola 1992, Texas 1997, Motorola 2000]. For this reason the research community has focused on this type of memory.

Multiple research efforts have explored the area of cache allocation for real-time systems. Here we mention just a couple that are close to our research. Avissar and Barua developed a scheme to optimally allocate global and stack variables to scratchpad statically [Avissar 2002]. They use profiling tools to gather information about the access frequency of the variables and transform the allocation problem into a 0/1 integer linear program for which they use Matlab, a commercially available software tool for numerical analysis, to solve it. Their approach, however, is not capable of handling linked data structures.

Dominguez, et al., developed a method to dynamically move linked data structures from heap into scratchpad [Dominguez 2005]. Their method avoids scanning and changing pointers in the data structures by ensuring that such pointers are only de-referenced when they are in the scratchpad and they are always placed on the same scratchpad location. They divide the program into regions with a single point of entrance regarding the control flow and insert in such points loading/unloading instructions.

Even though multiple research efforts have targeted the use of cache and scratchpad memory in embedded systems they mainly focus on a task-by-task approach. As a result, their optimization techniques have not been evaluated for system-wide impact. In particular, our approach utilizes the known periods of the tasks and their profiled access patterns to determine the most frequently accessed memory regions and allocation into cache (or scratchpad memory).

3.3 APPROACH

In this section we present our scheme for allocating cache memory to real-time task sets. This allocation is based on partitions where each task is given a fixed cache partition that improves the overall worst-case execution time of the task set. While this model was developed for caches it can also be applied to scratchpad memory by adjusting the parameters.

We extend the traditional Liu and Layland task model to include the demand of memory bandwidth where caches play a big role [Liu 1973]. In our task model, a task either accesses memory that was accessed in the past (repeat access) or memory that was not touched before. The amount of time a task accesses a specific segment of memory is captured as the fraction of the cycles needed to execute all the instructions of the task (if all the instructions were in the cache). This gives us a model as follows:

$$\tau = \left\lceil T, \left\{ G_i = \left(R_i, L_i, S_i, \frac{D_i}{C} \right) \right\} \right\rceil$$

where T is the period of the task and each triplet $G_i = (R_i, L_i, S_i, D_i/C)$ is the memory access in a segment of code from the set that constitutes the full task code. In each segment the parameters are as follows: R_i is the window of memory that is of repeated access (in bytes); L_i is the window of memory of linear access (in bytes); S_i is the amount of memory of random access that a read-ahead access would not be able to capture; D_i is the duration of the phase, i.e., the number of cycles needed to execute the instructions that access this memory segment; and C is the number of cycles needed to complete the whole activation of the task. Note that this form of a task shows all possible memory access patterns. However, for each memory segment it would be only one of R_i , L_i , or S_i and would have a non-zero value while the other would be zero.

The processor partition is characterized by the parameters: $\rho = \omega, \lambda, \alpha$, where ω is the cache memory size of this partition, λ is the read-ahead capacity of the cache, and α is the access time (in cycles) of the memory access per byte.

With this specification the mapping from one task τ to a partition ρ gives us a slowdown δ_i for each phase i due to memory access formulas follows:

$$\delta_i = \left\lceil \frac{\lambda_i - \lambda + \gamma_i}{\lambda} + \gamma_i \right\rceil \frac{x_i}{C}$$

Assuming tasks that are periodic where the code executed in each period is the same then the slowdown rate¹ per period per phase is: $\pi_i = \frac{\delta_i}{T}$. And the total slowdown rate of the task is:

$\Delta = \sum_{i \in \tau} \pi_i$. With this slowdown rate it is possible to perform an optimization of the cache allocation across all tasks to gain the maximum speed-up in the system.

We have adopted QRAM (Quality of Service Resource Allocation Model) as our allocation scheme [Rajkumar 1998, Chen 1999]. It optimizes the resource allocation based on its payoff. In other words, given multiple demands it allocates resources to the highest-paying demands. In the case of cache, the resource is the cache and the demands are the memory locations being accessed. The payoff of allocating one memory location (or window of locations) to the cache is the speed-up experienced by avoiding accessing the location in main memory. The demand in this case can be bundled as windows of memory segments that can be allocated to the cache. For each of these windows we then calculate the payoff as the access time we save per byte allocated to cache (that can be seen as a payoff or speedup rate). The windows are then put in an allocation vector in decreasing order of payoff so that the first window in the vector is the highest paying allocation possible. Thus, QRAM traverses the vector in order to allocate each window to the cache until it runs out of cache. The order of the vector is fundamental for the optimization of QRAM because it ensures that when QRAM allocates resources, and possibly exceeds them, the demands that are not covered always pay less than the ones allocated.

We calculate the allocation payoff, denoted γ_i (slope), for each memory segment of the task as

follows: $\gamma_i = \frac{\pi_i}{\lambda_i + \gamma_{i-1}}$. The payoff vector Γ for each task τ is derived as $\Gamma = \gamma_i$, where if $\gamma_i > \gamma_j$ then $i < j$.

QRAM then builds a speedup graph for each task τ as shown in Figure 3-1.

1 Calculating δ_i as if no cache were assigned, i.e., $\omega = 0$.

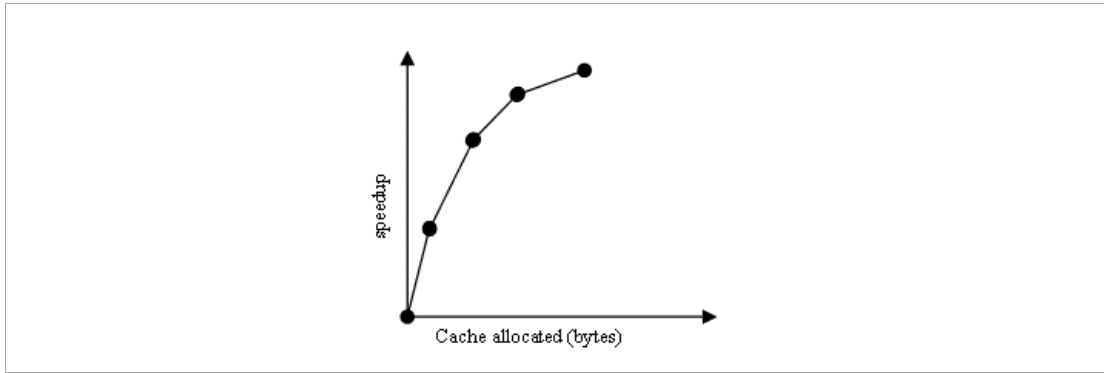


Figure 3-1: Speedup to Cache Allocation Function

In the case of multiple tasks, instead of having a single vector Γ , we have a collection of vectors Γ_i , one vector for each task τ_i . In this case QRAM keeps an index per vector that points to the next demand in the vector and selects, from all the next demands, the highest paying one (again highest speedup). Once this demand is chosen then its index is incremented. This way we are able to select the highest paying demand across multiple tasks. The iteration is repeated until all the cache is assigned. A sample assignment of two tasks is presented in Figure 3-2. In this figure the assignment sequence is presented as a label for each segment in the task.

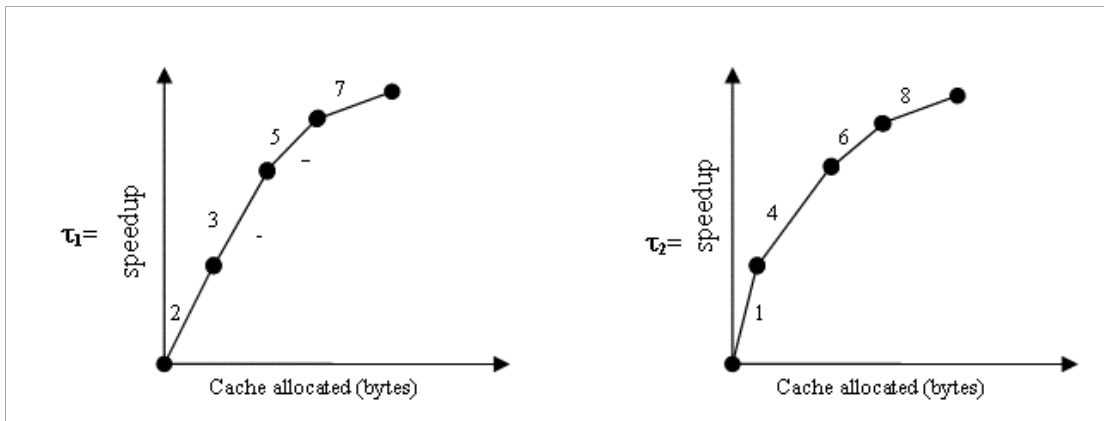


Figure 3-2: Two Task QRAM Cache Assignment

Discovering memory access patterns from executable code can be a complex task. A useful alternative to this approach is using memory access traces that can be gathered from processor simulators. Using memory access traces it is possible to evaluate the access frequency of each byte of code in the total trace. We denote this frequency the memory access density (MAD), which can then be used to gather the payoff of statically allocating each of the memory addresses to cache. This payoff in fact represents the speedup that accessing cache represents as compared with accessing to main memory. From the individual MAD per byte it is possible to group together contiguous memory locations into a window that in fact represents the memory windows as well as the payoff slope γ_i .

3.4 COLLABORATIONS

We have had fruitful discussions with Bill Milam at Ford.

3.5 EVALUATION CRITERIA

The evaluation criteria for this project were to

- develop a framework to improve the predictability of the timing behavior of real-time systems in the presence of caches
- implement a prototype for our scheme that can provide us with evidence of the effectiveness of our scheme
- conduct experiments with realistic cases that give us an indication of the degree of effectiveness of our result

3.6 RESULTS

Embedded systems practitioners rarely use cache because of the unpredictability it brings to the execution time. For this reason they usually disable the cache. However, as the speed gap between processor and memory is enlarged, the use of cache becomes very difficult to avoid. As a result, new techniques to use cache memory, while avoiding unpredictable execution time, have emerged. In this project we presented a near-optimal static allocation algorithm for cache memory (and other on-chip memory) that optimizes the speedup for memory. In particular, our algorithm can be used with memory access density (or the frequency of use of memory bytes) to represent the number of misses to the cache that can be saved if a particular memory location is allocated to cache. The multiplicity of tasks that embedded real-time systems have was also included in the algorithm by adding the periodicity of the tasks to the memory access density to get the final speedup. To perform the optimal allocation the Quality of Service Resource Allocation Model (QRAM) was used to perform the allocation across tasks. We have performed a series of experiments to evaluate our model and have shown the cases in which our model is more effective and in which they are not as effective. In particular, we have confirmed that when applications focus on a single execution “stage” (accessing a single memory window repeatedly), which is typical for embedded real-time systems, our technique is most effective. In contrast, our technique becomes less effective for programs with multiple stages. Finally, we have derived a worst-case pattern that showed us the worst-case over-allocation that must be conducted to gain comparable speed between a common cache replacement algorithm, least recently used (LRU), and static allocation algorithms. This pattern has shown us the specific benefits gained with the static allocation algorithms against dynamic algorithms when using them in embedded real-time systems.

In summary, in this context, our contribution is four-fold. First, we have shown how to use memory access density to evaluate the speedup that can be gained in a static allocation. Second, we have demonstrated how to use the QRAM allocation algorithm to add the periodicity of real-time tasks to the allocation scheme in a near-optimal fashion. Third, we have provided experiments that confirm the utility of our scheme. Further, we have described the worst-case pattern that would result in the upper bound of the worst-case execution under a dynamic replacement algorithm LRU.

3.7 PUBLICATIONS AND PRESENTATIONS

Dio De Niz, Peter Feiler, Jörgen Hansson, & John Hudak. “Near-Optimal Cache Partitioning for Real-Time Systems,” technical report. Software Engineering Institute, Carnegie Mellon University, 2008 (forthcoming).

3.8 REFERENCES

[Adieletta 2002]

M. Adieletta, M. Rosenbluth, D. Bernstein, G. Wolrich, & H. Wilkinson. “The Next Generation of Intel IXP Network Processors.” *Intel Technology Journal* 6, 3 (August 2002).
<http://developer.intel.com/technology/itj/2002/volume06issue03/>

[Avisar 2002]

Oren Avisar & Rajeev Barua. “An Optimal Memory Allocation Scheme for Scratch-Pad-Based Embedded System.” *ACM Transactions on Embedded Computing Systems* 1:1 (November 2002): 6-26.

[Brash 2002]

David Brash. “The ARM Architecture Version 6 (ARMv6).” ARM Ltd., January 2002. White Paper.

[Chen 1999]

Chen Lee, John Lehoczky, Ragunathan (Raj) Rajkumar, & Dan Siewiorek. “On Quality of Service Optimization with Discrete QoS Options.” IEEE Real-Time Applications and Systems Symposium, 1999.

[Dominguez 2005]

Angel Dominguez, Sumesh Udayakumaran, & Rajeev Barua. “Heap Data Allocation to Scratch-Pad Memory in Embedded Systems.” *Journal of Embedded Computing* 1, 4 (December 2005): 521-540.

[Edler 2003]

J. Edler & M. D. Hill. “Dinero IV.” 2003. <http://www.cs.wisc.edu/~markhill/DineroIV/>

[Liu 1973]

C. L. Liu & J. W Layland “Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment.” *JACM* 20, 1 (1973): 40-61.

[Motorola 1998]

M-CORE – MMC2001 Reference Manual. Motorola Corporation, 1998. (A 32-bit processor).
<http://www.ec66.com/market/sheet/MMC2001RM.pdf>

[Motorola 2000]

“CPU12 Reference Manual. Motorola Corporation (A 16-bit processor).” 2000.
http://www.freescale.com/files/microcontrollers/doc/ref_manual/CPU12RM.pdf

[Rajkumar 1998]

Ragunathan (Raj) Rajkumar, Chen Lee, John P. Lehoczky, & Daniel P. Siewiorek. "Practical Solutions for QoS-Based Resource Allocation Problems." IEEE Real-Time Systems Symposium, 1998.

[Sha 1994]

Lui Sha, Ragunatha Rajkumar, & Shirish S. Sathaye. "Generalized Rate-Monotonic Scheduling: A Framework for Developing Real-Time Systems." Proceedings of the IEEE, 82, 1 (January 1994).

[Texas 1997]

"TMS370Cx7x 8-bit Microcontroller." Texas Instruments, 1997.
<http://www-s.ti.com/sc/psheets/spns034c/spns034c.pdf>

4 A Research Agenda for Service-Oriented Architecture

Grace A. Lewis (Lead), Kostas Kontogiannis, Marin Litoiu, Stefan Schuster, and Dennis B. Smith

4.1 PURPOSE

It is clear that service-oriented architecture (SOA) is having a substantial impact on the way software systems are developed. However, although significant progress is being made on several fronts, current efforts seem to be evolving in many directions. There is a danger that important research needs will be overlooked, while other efforts will focus on issues of peripheral long-term significance in practice. As a research community that has gone through a substantial “growth spurt” we find ourselves facing a great opportunity and challenge: to better channel our research efforts, we should attempt to reflect upon our progress to date and recognize how our efforts and results build on each other, and to identify—and potentially prioritize—the areas that we still need to investigate. The purpose of this project is to provide a long-term consensus SOA research agenda, classified into research issues pertaining to the business, engineering, and operation aspects of service-oriented systems, plus a set of cross-cutting aspects. An additional goal is to assemble an international research group to analyze the current state of the practice and current research initiatives in SOA and create a community of interest around SOA research to share results and ideas on how to improve SOA adoption and the development of service-oriented systems.

4.2 BACKGROUND

Over the past decade we have witnessed a significant growth of software applications that are delivered in the form of services utilizing the network infrastructure. These services are available either on corporate intranets or on the internet, and are delivered either on open or proprietary network protocols. This approach to systems development is commonly referred to as service-oriented architecture, SOA-based systems, or service-oriented systems.

The initially slow but gradually increasing adoption of service-oriented systems is supported by both the technical and the business community. From a technical perspective, service-oriented systems are an approach to software development where services provide reusable functionality with well-defined interfaces; where a service infrastructure enables discovery, composition and invocation of services; and where applications are built using functionality from available services. From a business perspective, service-oriented systems are a way of exposing legacy functionality to remote clients, implementing new business process models by utilizing existing or third-party software assets, and reducing overall IT expenditures while potentially increasing the potential for innovation through software investments [Bieberstein 2006]. From either perspective, and despite their initially slow adoption and the conflicting standards proposed to support them, service-oriented systems are becoming the de-facto approach to bridging the gap between business models and software infrastructure, and flexibly supporting changing business needs [Marks 2006].

A group of European researchers has identified a list of research challenges in service-oriented computing (SOC) with a goal similar to that of this project: *to provide the means for consolidating and streamlining current SOC research efforts, as well as prioritizing important gaps* [Papa-

zoglou 2007]. While the results of this work are important and compatible with the results of this project, the work focuses largely on the development and operations of service-oriented systems. Our proposed taxonomy of research issues takes a step back and identifies issues that would help an organization make decisions on SOA adoption, create an SOA strategy, and then develop, deploy, and operate service-oriented systems.

4.3 APPROACH

The project started with an extensive literature review on topics related to SOA, with the purpose of identifying the state of the practice, plus multiple interviews with practitioners and researchers to identify both enablers of and barriers to SOA adoption. This literature review included multiple case studies of successful SOA adoption. Most of these case studies, though mostly vendor-sponsored and product-specific, all had a theme in common: a strong link between business strategy and SOA adoption. With this in mind, we created a service-oriented system development life cycle that supports the strategic approach to SOA adoption shown in the case studies. We then identified areas of SOA research necessary to fill in the gaps and developed a draft agenda, which we validated with a diverse community at seven international workshops and one international panel. A final report with our findings is scheduled for publication this year.

4.4 COLLABORATIONS

Within the SEI, the core project team consisted of Grace Lewis and Dennis Smith. We also received valuable input from Soumya Simanta. Outside the SEI, core team members were Kostas Kontogiannis from National Technical University of Athens, who provided an academic and practitioner perspective; Marin Litoiu from IBM Canada, who provided an industry perspective; and Stefan Schuster from the European Software Institute, who provided a European research and industry perspective. Hausi Müller from the University of Victoria and Eleni Stroulia from the University of Alberta also provided valuable input to the results.

4.5 RESULTS

4.5.1 Overview of the SOA Research Framework

In an ideal service-orientation adoption setting, an organization develops a service strategy that takes into account the organization's business drivers, context and application domain. In order to execute the service strategy, the organization has to generate plans to achieve the goals and objectives outlined by the strategy. Finally, the execution of these plans requires that business, engineering and operations decisions be made, taking into consideration cross-cutting concerns such as governance, social and legal issues, stakeholder management, and training and education. These relationships are shown in Figure 4-1.

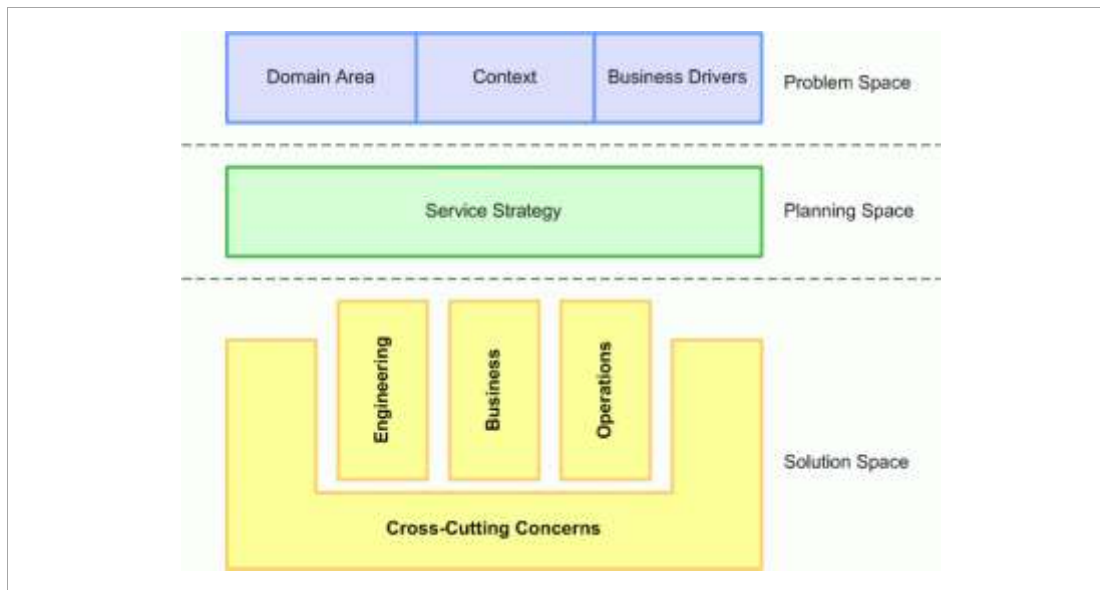


Figure 4-1: Overview of the SOA Problem and Solution Space

Problem Space: The problem space corresponds to the characteristics of the organization that will adopt SOA, as well as the problems that SOA is expected to address. The problem space shapes and places constraints on the strategy, but can also enable its execution. The elements of the problem space become the drivers for the strategy.

Planning Space: An SOA strategy should be stated as the way in which SOA will address the organization’s business drivers for SOA adoption. A service-oriented environment is iterative, to the point that the term “perpetual beta” is being used to indicate the dynamism of this environment in responding to requirements for business agility. The organization’s SOA strategy may change over time because of changes in the problem space or to information provided by data collected during evaluation/optimization, as shown in Figure 4-2.

Solution Space: In the solution space the SOA plans are executed to produce a service-oriented system. During execution, changes or wrong assumptions about SOA technology may invalidate the plans and cause the organization to reformulate its SOA strategy, as shown in Figure 4-2. Once the service-oriented system is deployed, measurements are gathered to support any metrics designed to test the effectiveness of the SOA strategy and the system itself. This data will help to optimize the SOA strategy, if needed, and also help the organization plan for the next iteration, once again reflecting the dynamic nature of service-oriented environments.

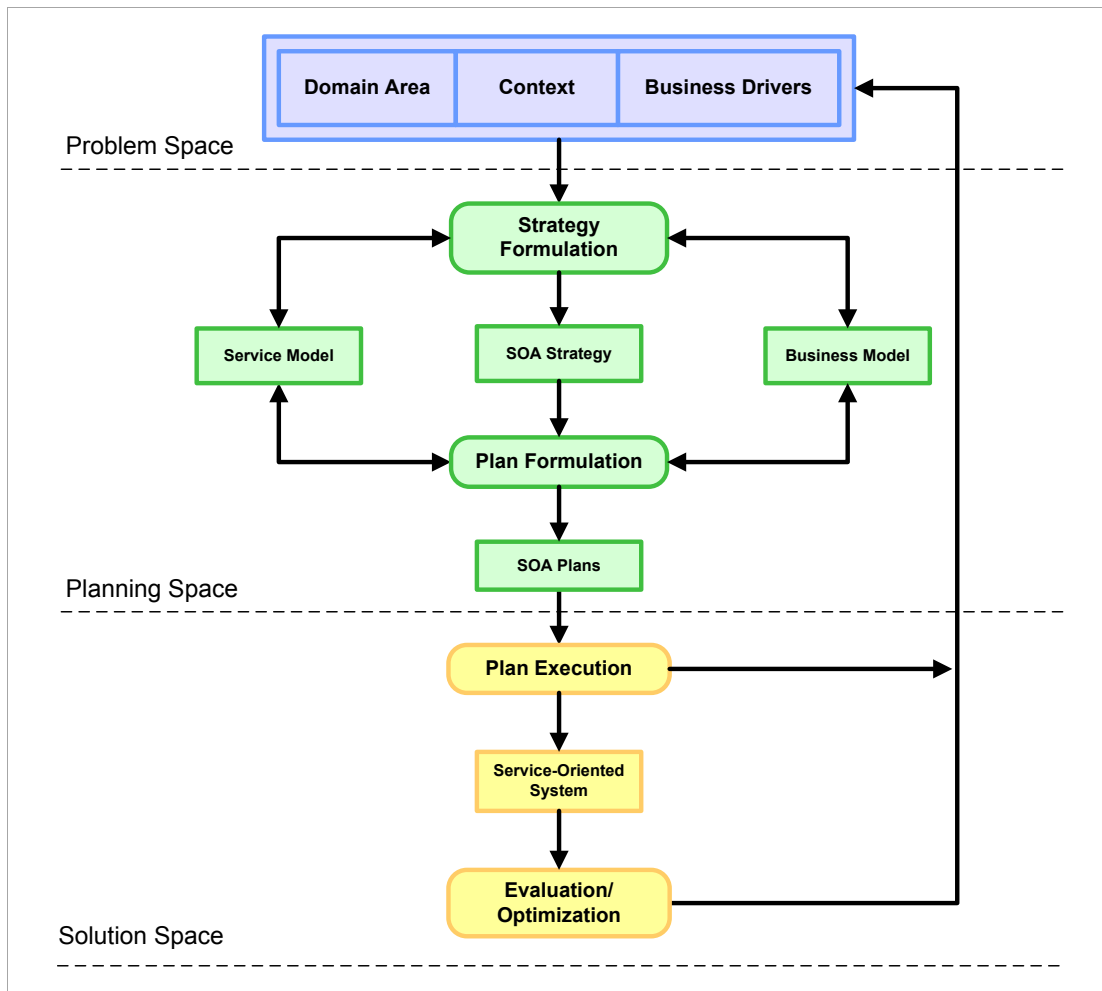


Figure 4-2: Expanded View of the SOA Problem and Solution Space

4.5.2 SOA Framework Phases, Activities, and Indicators

A strategic approach to SOA adoption requires an iterative approach to systems development that reflects the strong link between business strategy and development strategy. Figure 4-3 shows a proposed software development life cycle for service-oriented systems, where each pass through the life cycle corresponds to an iteration in Figure 4-2. The development phases are listed across the top and the activities that are carried out to develop the service-oriented system during these phases are on the left, along with indicators to evaluate the effectiveness of the service-oriented system against SOA adoption goals. The main differences with other iterative development frameworks, such as the IBM Rational Unified Process (RUP)², are the emphasis on activities to establish and analyze the relationship with business goals at the beginning of the cycle, the emphasis on evaluation at the end of the cycle, and the specification/review of business objectives at the end of the cycle so that the requirements for each iteration follow business objectives.

2 The IBM Rational Process website is <http://www-306.ibm.com/software/awdtools/rup/>

PHASES	P1: Analysis	P2: Positioning	P3: Assessment	P4: Planning	P5: Construction	P6: Transition	P7: Production
ACTIVITIES							
A1: Business Objectives Specification	+++	++	+				++
A2: Business Intelligence and Information Gathering		+++	+	++	+		
A3: Risk Analysis	++		+++	+++	++	+	
A4: Prototyping				++	+		
A5: Implementation				+	+++	++	
A6: Integration					++	+++	++
A7: Adoption						++	+++
A8: Maintenance					+	+	++
A9: Management						++	+++
INDICATORS							
I1: Financial Indicator Measurements	++		+	+++			
I2: Technology Indicator Measurements			+	++	++	+	++
I3: User Rating Measurements						++	+++
I4: Compliance Indicators	+			++	+	++	+++

Figure 4-3: Mapping Between Phases, Activities, and Indicators

4.5.3 SOA Research Taxonomy

The development of a service-oriented system requires business, engineering, and operations decisions to be made, as well as other cross-cutting decisions. Our taxonomy of research topics, shown in Figure 4-4, is divided into these decision areas. The research topics correspond to areas where additional research is needed to support a strategic approach to service-oriented systems development.

The complete final report will provide the rationale, current efforts, and challenges and gaps for each of the research topics identified in the taxonomy. Below we list examples of selected topics from each of the four primary areas of the taxonomy. Greater detail as well as an annotated bibliography for the sources of information will be included in the final report.

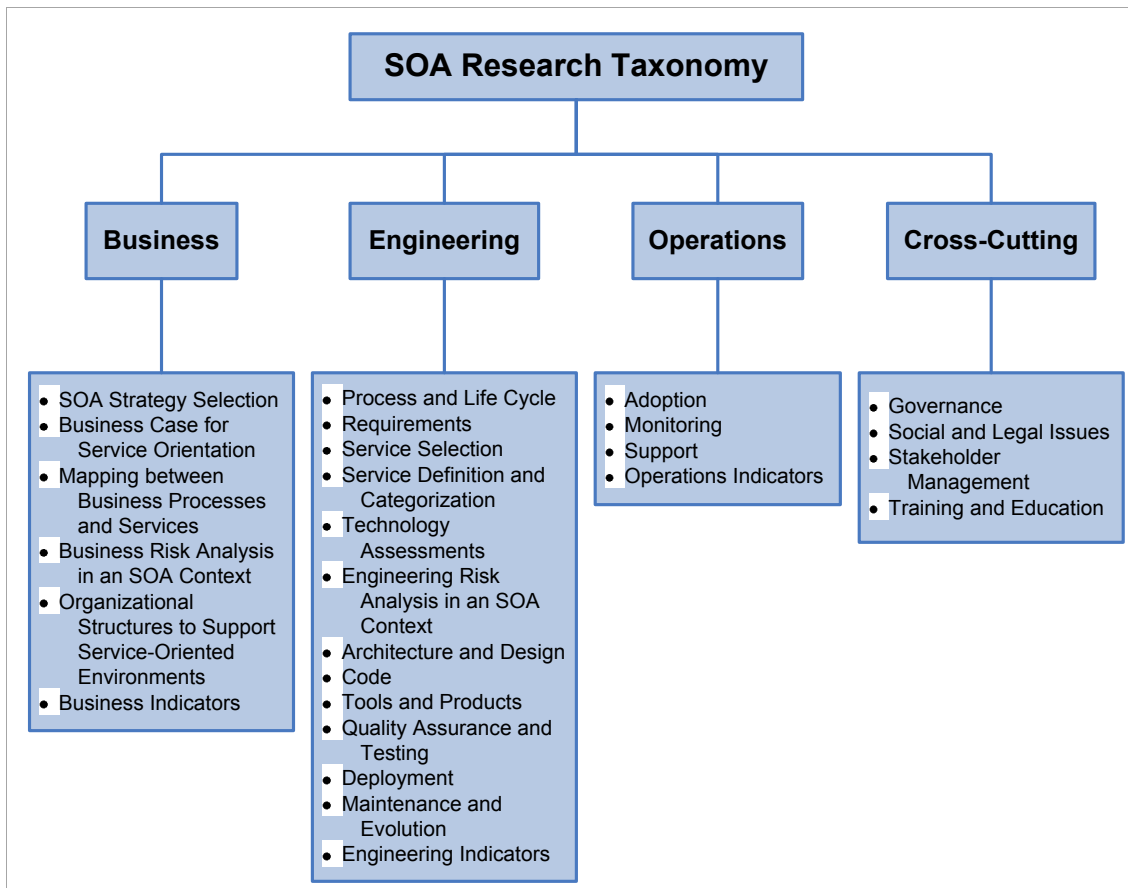


Figure 4-4: SOA Research Taxonomy

4.5.3.1 Example of a Business Topic: Business Case for SOA

Rationale: In general, there is recognition that SOA adoption can provide business agility, adaptability, legacy leverage, and integration with business partners. Given these goals, an important criterion for making business decisions concerns the amount of investment that is required for SOA adoption and the projected payoff over a certain period of time.

Current Efforts: There is current work that identifies the business value of SOA adoption in various industries, as indicated by references:

- Tilley, et al., discuss the business value of web services when used for enterprise application integration or business-to-business (B2B) commerce [Tilley 2004].
- Brandner, et al., claim enhanced integration capabilities of a core banking system through the use of web services [Brander 2004]. There are several other Australian, U.S., and Finnish success stories in the banking industry.
- Pujari discusses the pros and cons of self-service technology, potentially enabled by SOA technologies, in the Canadian B2B industry [Pujari 2004].
- Linthicum has written several articles on the subject, including one in which he proposes a formula for calculating the relative value of SOA adoption [Linthicum 2006].

There are other case studies and articles that provide anecdotal evidence of the business value of SOA adoption—many of these studies are sponsored by vendors or co-authored with vendors. The problem is that these are examples of point solutions that are all so different that it is difficult to make valid generalizations across organizations. A comprehensive framework for understanding the business value of SOA has not yet been developed.

Challenges and Gaps: The largest gap in this area is the lack of vendor-neutral data. Current efforts have focused on individual case studies and there have not been rigorous analyses that can be generalized. This triggers another important question that has to do with the nature of the data to be gathered. How does an organization measure the common benefits associated with SOA adoption, such as business agility, legacy leverage, or increased interoperability? An important research focus would be to gather data from both success stories and failures, find commonalities, and start to develop a framework for calculating the business value of SOA adoption.

4.5.3.2 Example of an Engineering Topic: System Testing

Rationale: In an SOA environment, system testing means end-to-end testing. The problem is that in SOA environments, system components are distributed, deployed on heterogeneous platforms, and often not even available.

Current Efforts: The market for testing tools for SOA environments (mainly for web services) is growing. Tools are available to perform testing at multiple levels—from business processes to messages—as well as for qualities such as availability, performance, and security. However, most testing tools are incapable of building composite interdependent tests across technology platforms, languages, and systems. Also, most testing tools assume control over all elements of the service-oriented system. Sometimes client developers only have access to interfaces (e.g., WSDL description files in the case of web services) and lack access to code. This has triggered some research into the use of gray-box³ testing, which is appropriate when there is limited knowledge.

Challenges and Gaps: The challenges in system testing are driven by the distributed, heterogeneous nature of service-oriented systems components and a growing market of third-party services, which means that there is no single owner of the complete system. This triggers some interesting research topics, such as

- dynamic testing in distributed, heterogeneous environments
- service certification
 - What does a certification process look like?
 - What can be certified?
- enhanced service repositories that provide test cases for services
 - How are test cases specified?
- test-aware interfaces for service consumers to test services
 - Given that providers would need to have test instances of services, how are these test services specified and how do service consumers become aware of their existence?

³ Gray-box testing, according to Wikipedia, “involves having access to internal data structures and algorithms for purposes of designing test cases, but testing at the user, or black-box level.” See http://en.wikipedia.org/wiki/Software_testing.

We also need to recognize that it is not always possible to do end-to-end testing. In such cases, interesting research topics are

- simulation of service-oriented system environments
- best practices for exception handling

4.5.3.3 Example of an Operations Topic: Service-Level Agreements

Rationale: A service-level agreement (SLA) is a formal and bilateral contract between a service provider and a consumer to specify the requirements and uses of specific services. An SLA is essential for establishing trust between service providers and service consumers. In a growing third-party service market, the establishment of SLAs can be used to differentiate and select from various available services and to help service providers anticipate demand and plan their resource allocation accordingly. They can also be used as a mechanism for risk mitigation.

Current Efforts: There is a perceived need for standardization, specification, and guidance for using SLAs in an SOA context. The web service level agreement (WSLA) is a specification and reference implementation by IBM that provides detailed SLA specification requirements for enabling the monitoring of SLA compliance, describes how these requirements are addressed in the WSLA specification, and provides a WSLA monitoring framework that allows monitoring of SLAs at runtime [IBM 2007]. CBDi Forum provides basic guidance on how to approach SLAs from the service consumer and service provider perspectives at a higher level than WSLA [CBDi 2006]. Given that most SLAs are based on specifying the required quality of service (QoS) for a service, an active research area is modeling and implementing various QoS attributes in service-oriented and dynamic environments.

Challenges and Gaps: An important contribution to SOA adopters would be the creation of a generic and standardized framework for SLA management across enterprises as well as across various lines of business inside an organization. This would involve providing appropriate automation and support for mapping contractual service level agreements to standard and actionable implementations and the monitoring and management of service level at runtime. In the area of QoS, more work needs to be done in understanding QoS of composite services, especially when lower level services in a composite service are provided by different providers.

4.5.3.4 Example of a Cross-Cutting Topic: SOA Governance

Rationale: An InfoWorld 2007 SOA Trend Survey indicates that lack of governance is the main inhibitor for SOA adoption (50%). Effective SOA governance requires rules that define roles and responsibilities, define appropriate use of standards, make explicit the expectations of a diverse set of stakeholders, provide for SLAs, and monitor compliance through metrics and automatic recording and reporting.

Current Efforts: A number of organizations such as IBM, AgilePath, and Software AG have developed sophisticated models of SOA governance. These models focus mostly on relationship to corporate enterprise architecture, use of registries, management of SOA life cycles, how to define and monitor SLAs, and how to define and analyze metrics on policy enforcement, effectiveness of services, and use of services. A number of tools have also begun to automatically incorporate me-

trics and aspects of governance, and research efforts have begun to identify roles and responsibilities [Kajko-Mattsson 2007, Kajko-Mattsson 2008].

Challenges and Gaps: Most efforts to define and implement governance are still vendor driven and guided by the governance aspects that can be automated by their tools. As with the business case for SOA, most case studies are anecdotal and idiosyncratic. An interesting research topic would be to establish an abstract model for SOA governance and its variations within different domains. A starting point could be the establishment of SOA governance elements, similar to what has been done at Hartford Inc. with the creation of a template. This is similar to work done by Burton Group [Afshar 2007, Manes 2007].

4.5.4 Results and Future Work

Engineering challenges are significant if SOA is to be used in advanced ways, such as semantic services, dynamic discovery and composition, and real time applications. The main challenges for enterprise applications are related to business and operations, and not engineering. As third-party services become the new business model, we will need support for service-level agreements, run-time monitoring, end-to-end testing involving third parties, pricing models for third-party services, and service usability from a design and adoption perspective.

In some areas, non-vendor surveys, studies, and experiments are needed to produce more concrete guidance, rather than additional basic research. Some examples of these areas are SOA governance, a business case for SOA adoption, return on investment for SOA adoption, and development processes and practices for SOA-based development.

We also found several topics in which there is significant research in academia, such as semantics, but no support from industry to test ideas in real scenarios. We need more collaborative research between industry and academia to create real practices.

The next steps for this project are to create a final report with the complete findings and to start establishing a community of interest around the SOA research agenda.

4.6 PUBLICATIONS AND PRESENTATIONS

The development of the SOA Research Agenda has led to the following publications (and corresponding presentations at the workshops):

- Kontogiannis, K., Lewis, G., & Smith, D. “The Landscape of Service-Oriented Systems: A Research Perspective for Maintenance and Reengineering.” Proceedings of the International Workshop on Service-Oriented Architecture Maintenance and Reengineering (SOAM 2007). International Conference on Software Maintenance and Reengineering (CSMR 2007), March 2007.
- Kontogiannis, K., Lewis, G., Litoiu, M., Muller, H., Schuster, S., Smith, D., & Stroulia, E. “The Landscape of Service-Oriented Systems: A Research Perspective.” Proceedings of the International Workshop on Systems Development in SOA Environments (SDSOA 2007). International Conference on Software Engineering (ICSE 2007), May 2007.

Additionally, the work was presented at the following conferences:

- IBM Center for Advanced Studies Conference (CASCON 2006): Workshop on the Effects of Service-Oriented Architecture on the Software Development Life Cycle, October 2006.
- International Conference on Composition-Based Systems (ICCBSS 2007): Panel “A Research Agenda for SOA,” March 2007.
- European Conference on Software Maintenance and Reengineering (CSMR 2007): SOAM 2007—Workshop on Service-Oriented Architecture Maintenance, March 2007.
- International Conference on Interoperability for Enterprise Software and Applications (I-EISA 2007): FSOA 2007—Foundations of Service-Oriented Architecture, March 2007.
- Consortium for Software Engineering Research (CSER) Spring Meeting: Workshop “A Research Agenda for SOA,” April 2007.
- International Conference on Software Engineering (ICSE 2007): SDSOA 2007—Workshop on Systems Development in SOA Environments, May 2007.
- International Conference on Software Maintenance (ICSM 2007): MESOA 2007—Workshop on Maintenance and Evolution of SOA-Based Systems, October 2007.
- CASCON 2007: Workshop “SOA Research Challenges: A User Perspective,” October 2007.

4.7 REFERENCES

[Afshar 2007]

M. Afshar & B. Moreland. “Keys to Successful Governance with SOA.” Presentation at the Transformation and Innovation 2007 Conference, May 2007.

[Bieberstein 2006]

Norbert Bieberstein. *Service-Oriented Architecture Compass—Business Value, Planning and Enterprise Roadmap*. Upper Saddle River: Pearson, 2006 (ISBN: 0131870025).

[Brandner 2004]

M. Brandner, M. Craes, F. Oellermann, & O. Zimmermann. “Web Services-Oriented Architecture in Production in the Finance Industry.” *Informatik-Spektrum* 27, 2 (April 2004): 136-145.

[CBDi 2006]

CBDi. “Service Level Agreements: Best Practice Report.” *CBDi Journal*. December 2006.

[Fitzgerald 2006]

B. Fitzgerald & C. M. Olsson (eds). “The Software and Services Challenge. Contribution to the Preparation of the Technology Pillar on Software, Grids, Security and Dependability.” EY 7th Framework Programme, 2006.

[IBM 2007]

Web Service Level Agreements (WSLA) Project. <http://www.research.ibm.com/wsla/> (2007).

[Kajko-Mattsson 2007]

Mira Kajko-Mattsson, Grace Lewis, & Dennis Smith. “A Framework for Roles for Development, Evolution, and Maintenance of SOA-Based Systems.” Proceedings of the International Workshop on Systems Development in SOA Environments (SDSOA 2007). International Conference on Software Engineering (ICSE 2007), May 2007.

[Kajko-Mattsson 2008]

Mira Kajko-Mattsson, Grace Lewis, & Dennis Smith. “Evolution and Maintenance of SOA-Based Systems at SAS.” Proceedings of the 41st Hawaii International Conference on System Sciences (HICSS-41), January 2008.

[Linthicum 2006]

D. Linthicum. “When Building a SOA, How Do You Know When You're Done?” *InfoWorld*, June 2006.

http://weblog.infoworld.com/realworldsoa/archives/2006/06/when_building_a.html

[Manes 2007]

A. T. Manes. “SOA Governance Infrastructure.” Burton Group, 2007.

[Marks 2006]

E. Marks & M. Bell. *Service Oriented Architecture: A Planning and Implementation Guide for Business and Technology*. Hoboken: John Wiley & Sons, 2006 (ISBN: 0471768944).

[Papazoglou 2007]

M. Papazoglou, P. Traverso, S. Dustdar, & F. Leymann. “Service-Oriented Computing: State of the Art and Research Challenges.” *IEEE Computer* 40, 11, (November 2007): 38-45.

[Pujari 2004]

D. Pujari. “Self-Service with a Smile? Self-Service Technology (SST) Encounters among Canadian Business-to-Business.” *International Journal of Service Industry Management* 15, 2 (2004): 200-219.

[Tilley 2004]

S. Tilley, J. Gerdes, J. T. Hamilton, S. Huang, H. A. Müller, D. Smith, & K. Wong. “On the Business Value and Technical Challenges of Adopting Web Services.” *Journal of Software Maintenance and Evolution: Research and Practice* 16, 1–2 (January–April 2004): 31–50.

5 A Software System Engineering Approach for Fault Containment

Peter H. Feiler, Dionisio de Niz, Jörgen Hansson, Lui Sha, and Don Firesmith

5.1 PURPOSE

Why do system-level failures still occur despite the deployment of fault-tolerance techniques in systems? A lack of effective system-level fault management and stability solutions, despite best efforts at fault tolerance, presents major challenges in modern avionics and aerospace. System engineering approaches in the form of hardware redundancy for managing hardware failures are well established. Providing a software system engineering approach for systematic fault management with predictable results remains a challenge. The following examples illustrate the point.

After years of development, F-22 flight tests began in late 1997, but the aircraft still experienced serious avionics instability problems as late as 2003. According to testimony from the U.S. General Accounting Office (GAO), “The Air Force told us avionics have failed or shut down during numerous tests of F/A-22 aircraft due to software problems. The shutdowns have occurred when the pilot attempts to use the radar, communication, navigation, identification, and electronic warfare systems concurrently” [Li 2003]. As shown in this example, the workload generated by different system configurations affects the execution characteristics of the application in ways that are difficult to trace. The workload may introduce instability due to violation of assumptions made by application software components about the timing and fault characteristics of the application data streams they operate on, which can lead to a failure.

In another example, the Ariane 5 rocket exploded during her maiden flight. The destruction was triggered by the overflow of the horizontal velocity variable in a reused Ariane 4 software component to perform a function that was “not required for Ariane 5” [Ariane 2008]. That is, a legacy feature that was not even needed destroyed the rocket. This is a dramatic example of system instability: failure due to inconsistent system configuration. A fault in an unneeded function was not contained and cascaded into a total system failure. The reason for the overflow was the representation of a vertical velocity value as a 16-bit integer—placing a range restriction on the value that was exceeded by Ariane 5. It could and should have been a minor fault that would have no impact on the flight, had the fault in the unneeded function been contained there.

The objectives of this project are to

- identify system fault behaviors that are not addressed by component-fault containment techniques
- develop a formalized analysis framework for system fault containment and stability management
- validate system architectures in the context of this framework

The focus of this framework is on system-level consistency characteristics and rules for identifying direct or indirect contributions to their violation by individual components and by infrastructure services. By extending best practices, including architecture modeling and analysis, architec-

ture patterns, component and system-level fault tolerance, and design rules, we will provide developers with what is required to place future developments on a sound theoretical footing. The project is a two-year project and this section summarizes the results of the first year.

5.1.1 Background

The National Coordination Office for Networking and Information Technology Research and Development (NITRD), in its work on high-confidence software and systems, has identified five technology goals that must be met to realize the vision of high-confidence software systems:

(i) Provide a sound theoretical, scientific, and technological basis for assured construction of safe, secure systems. (ii) Develop hardware, software, and system engineering tools that incorporate ubiquitous, application-based, domain-based, and risk-based assurance. (iii) Reduce the effort, time, and cost of assurance and quality certification processes. (iv) Provide a technology base of public domain, advanced-prototype implementations of high-confidence technologies to enable rapid adoption. (v) Provide measures of results [NITRD 2001].

Virtual machines have been recognized as a key concept for providing robustness through fault containment in integrated modular avionics systems. Known as partitioned architecture in the avionics systems community, this mechanism provides time and space partitioning to isolate application components and subsystems from affecting each other due to sharing of resources. This architecture pattern can be found in the ARINC 653 standard [ARINC 2008]. In a recent study of the migration of an avionics system from a federated system architecture to a partitioned system architecture the SEI team has identified sources of previously absent system-level faults due to different age characteristics of data streams under the partitioned system runtime architecture [Feiler 2004].

Dr. Vestal from Honeywell has demonstrated that impact analysis based on models of the runtime architecture, that is, the application system deployed on an execution platform, can be the basis for isolation analysis and fault propagation modeling. Through error model and fault occurrence annotations, he demonstrated the feasibility of reliability and fault tree analysis from the same architecture model that was the basis of global schedulability analysis [Binns 2004]. His experience has led to incorporating the concept of error propagation into the error model annex of the SAE AADL standard. Similarly, the DARP initiative at York University has utilized architecture dependency information to perform fault propagation analysis [Wallace 2005].

Prof. Welch in his DeSiDeRaTa work has investigated a scalable resource management approach for distributed real-time systems in the context of the DD(X) program [Welch 1998]. His approach focuses on managing the desirable performance characteristics of critical information flows in a distributed embedded application system. The requirement for support of end-to-end flow specifications in support of system-level consistency analysis has been raised by the Future Combat System (FCS) system architecture contractor and other avionics and aerospace contractors.

Prof. Sha, while a member of the technical staff at the SEI, investigated an innovative approach to managing software fault tolerance in light of dependable system upgrade. This approach overcomes shortcomings of redundancy by replication through an analytically redundant fault container mechanism for software components that are control-system applications (Simplex) [Sha 1998]. In a collaborative project funded by the Defense Advanced Research Projects Agency (DARPA)

this technology was applied to an avionics system. (Researchers were Prof. Lehoczky, Prof. Rajkumar, and Prof. Krogh from Carnegie Mellon University, Prof. Sha and Dr. Feiler from the SEI, and Jon Preston from Lockheed Martin.) In the context of this project it was recognized that component-level fault containment can still lead to system-level inconsistencies that result in faulty behavior of other components. Under the guidance of Dr. Feiler, Jun Li investigated in a Ph.D. thesis the feasibility of capturing relevant characteristics of component interactions that would lead to system-level inconsistencies [Feiler 1998].

The PERFORM group at the University of Illinois, led by Prof. William H. Sanders, conducts research in the design and validation of dependable and secure networked systems. Such systems often have requirements for high performance, dependability, and security, and these goals may contradict one another. By providing a unified method to validate system performance, dependability, and security during the entire design process, the group develops and applies sound engineering principles to large-scale system design, advanced modeling, analysis, and simulation environments [Deavours 2002].

5.2 APPROACH

We have divided the project into three phases:

- root cause identification of system-wide fault propagation
- development of analytical frameworks to predict the impact of seemingly minor faults on the system operation
- development of architecture design guidance to reduce such faults

Root cause identification involved identification of high-priority and high-criticality system failures due to unexpected fault propagation and the factors that contribute to such failures. This activity draws on Lockheed Martin's experience with several fighter aircraft developments, in particular the F-16, F-22, and F-35. In 2008, an evaluation of problem history data from the Carnegie Mellon team for the DARPA Urban Grand Challenge robotic vehicle competition will be done.

The development of an analytical approach focuses on identifying inconsistencies of expectations in signal stream characteristics that can lead to system failures. This analytic approach leads to tool-based validation of system configuration consistency in architecture models and to fault containment rules that can be verified. The analysis is expected to incorporate each of the root cause contributors to the violation of these signal stream characteristics, for example, to the increase in latency variation, which can lead to unexpected control instabilities.

The insights from the analytical framework allow us to specify and analyze architecture patterns that are aimed at addressing robustness and stability in systems. Initial patterns being examined are the virtual machine/partitioned systems architecture, redundancy, and pipeline patterns. The specification includes the description of the problems it is intended to address and the understood consequences of using such a pattern. The analysis includes the identification of potential impact of the various runtime mechanisms of this execution platform for task execution and communication on those characteristics of critical application data streams that affect the robustness and stability of the system.

We have chosen the Architecture Analysis and Design Language (AADL) standard as a basis for these analysis frameworks for model-based engineering because of its strength to be (1) non-ambiguously and objectively human readable *and* (2) processable and analyzed by machines due to well-defined semantics. This system-level approach to fault containment has significant technical and programmatic merits that complement those of a component-level fault containment approach. Technically, system stability is achieved by a combination of component-level fault containment and ensuring well-formed dependency at the system level. Component-level fault containment ensures the safe sharing of hardware and logical services. That is, a component's faults cannot corrupt other components' code and data, and cannot overuse its CPU quota; nor can a component's faults corrupt the common OS and middleware services. An AADL-based system engineering approach enforces design rules for well formed dependency, meaning that we can verify that a component may use but not depend on the service of a less critical component. Well-formed dependency is a key to system dependability as it prevents a minor fault cascade into a major failure.

5.3 COLLABORATIONS

During the first year we have utilized the existing collaboration between Lockheed Martin Corporation and Prof. Sha, and GrammaTech. GrammaTech has loaned, free of charge, its CodeSonar tool to Prof. Sha to support experimentation. In addition, Prof. Sha has several doctoral students conducting related research who provide assistance. During the second year, we are going to continue these successful collaborations.

For the second year we will also collaborate with Prof. Raj Rajkumar of the Department of Electrical and Computer Engineering at Carnegie Mellon University, with a focus on the experience of Carnegie Mellon's victorious robotic car for the Urban Grand Challenge and the development and validation of our defect-prevention scheme in this platform. We believe the experience of the Grand Challenge is an important complement to the F-22/F-35 projects by providing a different application domain with a multiplicity of time-sensitive environment observations. In addition it will provide a readily available validation platform.

5.4 EVALUATION CRITERIA

The key criteria for evaluating this project have been the ability to identify several root cause areas of system-wide fault propagation, to develop or adapt existing frameworks to predictably identify contributors to those root causes through analysis of architecture models, and to codify guidance in architecture patterns.

In addition, our objective is to apply the results to an actual project and make them the foundation for methods and tools that can be used to support the F-35 and other programs.

5.5 RESULTS

5.5.1 Root Cause

During the first year we have identified four root cause areas of system-wide faults that are not addressed by traditional fault tolerance techniques:

- *Exclusive resource use assumptions in partitioned architectures:* Partitioned architectures, as promoted by the ARINC653 standard, offer a virtual processor concept that provides both time and space partitioning—giving the illusion of exclusively dedicated hardware. Large-scale embedded applications can be modularized into partitions, known as Integrated Modular Avionics (IMA) in the avionics domain, and deployed on a range of distributed computer platforms. Experience with actual systems has shown that use of the partition concept can still lead to performance issues due to unplanned resource sharing across partitions. We have characterized several actual problem scenarios in the use of partitions that have been encountered with the F-35 through the use of AADL models. This allowed us to pinpoint the key contributors to the unexpected reduction of performance in both the application and in the runtime infrastructure.
- *Mismatched data stream assumptions:* Control engineers make assumptions about the physical systems that are being observed and controlled. They create models of their algorithms and the controlled systems at various fidelity levels. These models are analyzed to gain confidence in the stability of the system. Application developers translate these control equations into application software components that execute in a real-time system environment in discrete time, in many cases distributed across multiple processors. These implementation choices affect the assumptions made by control engineers about the latency, latency jitter, and age of the data processed by the control loop, resulting in unexpected instability of the control behavior. We have identified a number of contributors to end-to-end latency variations due to choices in the implementation of embedded systems in software that result in potential control system instability.
- *Non-determinism in signal stream processing due to concurrency:* Modern avionics architectures are multi-threaded to increase utilization of individual processors through techniques such as rate-monotonic analysis and to take advantage of concurrency in distributed and multiple processor hardware platforms. This concurrency can lead to increased variation in signal data latency, which results in sporadic controller instabilities. We have identified a number of actual problems that are due to ordering assumptions made by embedded application code with respect to task execution and communication. Migration from legacy systems to multi-threaded, partitioned, and distributed architectures introduces concurrency, which if not managed properly results in unexpected non-deterministic behavior. Consequences of such behavior may result in control instabilities as well as unplanned stress on the physical systems.
- *Multiple time reference points:* Control system and mission system processing is time sensitive. During our analysis of avionics systems we have identified virtualization of timelines when migrating to partitioned systems as a second contributor to the non-deterministic signal stream processing behavior. This virtualization of timelines, when used by communication mechanisms, results in multiple independent time reference points (clocks). The use of glo-

bally asynchronous locally synchronous (GALS) architectures in some avionics systems, such as the F-22, has the same effect. Recent interactions with Prof. Rajkumar have confirmed that multiple time reference points are a major system fault root cause area in autonomous vehicles, such as those used in the DARPA Urban Grand Challenge.

During the first year we have developed two proof-of-concept analysis frameworks and have extended a third analysis framework.

- We have developed a fault impact analysis framework to model and validate fault propagation. This framework is based on and extends the fault propagation calculus (FPC) developed by Dr. Wallace at University of York [Wallace 2005]. We have mapped FPC into AADL and have developed a prototype analysis tool. This tool extends the original work by providing traceability between fault sources and impacted system components. In addition, the original calculus does not take into account the hardware platform, the partitioned architecture concept, and architecture dynamics. We have identified an approach to address those in the AADL-based fault impact analysis framework and plan to prototype it during year two. A prototype exists of the initial fault impact analysis capability and fault traceability support.

We have developed a fault-detection scheme that explores potential faults due to variations on concurrency (addition of threads and processors) that can be introduced in the system. We have demonstrated the feasibility of using chaotic system theory and model checkers, such as Alloy, as a computer-based solution. We have applied chaotic system theory developed by Ortmeier, et al. [Ortmeier 2004] as a way of exploring execution and communication ordering issues due to the introduction of concurrency to several examples from the avionics and automotive domains. We have utilized the AADL annex concept to extend AADL in support of concurrency constraint specifications. We have chosen Alloy [Jackson 2000] as a temporal specification language and have interfaced the Alloy toolset with the Open Source AADL Tool Environment (OSATE) to automatically generate Alloy models and identify counter examples—evidence of potential concurrency issues. A prototype of the integration of OSATE with Alloy has been made available to Rockwell-Collins and the University of Illinois at Urbana-Champaign.

- We have investigated the extension of a flow latency analysis framework to accommodate partitioned architectures and to account for latency jitter. Dr. Feiler originally developed a latency analysis capability in the context of AADL to demonstrate the potential of end-to-end flow analysis in embedded system architectures under a previous SEI research project. In the context of this project we have extended this analysis framework to take into account partitions, such as ARINC653 partitions, in its determination of maximum end-to-end flow latency. A prototype of this analysis capability is available as part of the OSATE toolset. We have identified the extensions to this analysis framework to also accommodate latency variation (jitter) analysis.

We have prototyped templates for codifying design guidance through architecture patterns. We have identified three key patterns that will be used in year two: the redundancy pattern, the partition pattern, and the pipeline pattern.

5.6 PUBLICATIONS AND PRESENTATIONS

Presentations have been given in a number of forums including the UIUC AADL workshop (December 2006), the SAE AADL Standards User Group meeting (January 2007, July 2007), the Open Group RT-Forum Workshop (January, April, July 2007), the International Workshop on Aspect-Oriented Modeling (March 2007), the Army Advisory Group (October 2007), the ARTIST2 Network of Excellence on Embedded Systems Design Workshop on Integrated Modular Avionics (November 2007), and the International Congress on Embedded Real-Time Systems (January 2008).

The following are publications related to this project with additional articles and technical reports in progress.

[de Niz 2007]

Dio de Niz & Peter H. Feiler. “Aspects in the Industry Standard AADL.” *Proceedings of 10th International Workshop on Aspect-Oriented Modeling*. Vancouver, Canada, March 2007.

[de Niz 2008]

Dio de Niz. “Architectural Concurrency Equivalence with Chaotic Models.” *5th International Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, 2008.

[Feiler 2007a]

Peter H. Feiler. “Integrated Modular Avionics: The Good, The Bad, and The Ugly.” ARTIST2 Network of Excellence on Embedded Systems Design Workshop on Integrated Modular Avionics, Proceedings, 2007. <http://www.artist-embedded.org/artist/Integrated-Modular-Avionics.html>

[Feiler 2007b]

Peter H. Feiler & Jörgen Hansson. *Flow Latency Analysis with the Architecture Analysis and Design Language (AADL)* (CMU/SEI-2007-TN-010). Software Engineering Institute, Carnegie Mellon University, 2007. <http://www.sei.cmu.edu/pub/documents/07.reports/07tn010.pdf>

[Feiler 2008]

Peter H. Feiler & Jörgen Hansson. “Impact of Runtime Architectures on Control System Stability.” *Proceedings of 4th International Congress on Embedded Real-Time Systems*, 2008.

5.7 REFERENCES

[Ariane 2008]

“Ariane 5 Flight 501.” http://en.wikipedia.org/wiki/Ariane_5_Flight_501

[ARINC 2008]

Avionics Application Software Standard Interface. “ARINC 653 Standard Document.” <http://www.arinc.com>

[Binns 2004]

P. Binns & S. Vestal. “Hierarchical Composition and Abstraction in Architecture Models.” IFIP TC-2 Workshop on Architecture Description Languages (WADL), World Computer Congress,

Aug. 22-27, 2004, Toulouse, France, Series: IFIP International Federation for Information Processing, Vol. 176, 2005 (ISBN: 0387245898).

[Deavours 2002]

D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derasavi, J. Doyle, W. H. Sanders, & P. Webster. “The Möbius Framework and its Implementation.” *IEEE Transactions on Software Engineering* 28,10 (2002): 956–970.

[Feiler 1998]

P. H. Feiler & J. Li. “Managing Inconsistency in Reconfigurable Systems.” *IEEE Proceedings Software* (1998): 172-179.

[Feiler 2004]

P. H. Feiler, D. P. Gluch, J. J. Hudak, & B. A. Lewis. “Pattern-Based Analysis of an Embedded Real-time System Architecture.” IFIP TC-2 Workshop on Architecture Description Languages (WADL), World Computer Congress, August 22-27, 2004, Toulouse, France, Series: IFIP International Federation for Information Processing, Vol. 176, 2005 (ISBN: 0387245898).

[Jackson 2000]

Daniel Jackson, Ian Schechter, & Ilya Shlyakhter. “Alcoa: the Alloy Constraint Analyzer.” *Proceedings of the International Conference on Software Engineering*, Limerick, Ireland, June 2000.

[Li 2003]

Allen Li. Testimony by Allen Li, Director, Acquisition and Sourcing Management, U.S. General Accounting Office, to the House Subcommittee on Tactical Air and Land Forces, Committee on Armed Services, April 2003. <http://www.gao.gov/new.items/d03603t.pdf>

[NITRD 2001]

NITRD—National Office for Networking and Information Technology Research, High Confidence Software and Systems Coordinating Group. “High Confidence Software and Systems Research Needs.” January 2001. <http://www.nitrd.gov/pubs/hcss-research.pdf>

[Ortmeier 2004]

F. Ortmeier, A. Thums, G. Schellhorn, & W. Reif. Combining Formal Methods and Safety Analysis: The Formosa Approach. Integration of Software Specification Techniques for Applications in Engineering. Part V: Verification. Lecture Notes in Computer Science (2004): 474-493.

[Wallace 2005]

Malcolm Wallace. “Modular Architectural Representation and Analysis of Fault Propagation and Transformation.” *Proceedings of Formal Foundations of Embedded Systems and Component-Based Software Architectures (FESCA)*, April 2005.
<ftp://ftp.cs.york.ac.uk/pub/malcolm/fesca05.html>

[Welch 1998]

L. Welch, B. Shirazi, & B. Ravindran. “DeSiDeRaTa: QoS Management Technology For Dynamic, Scalable, Dependable, Real-Time Systems.” *Proceedings of the 15th Symposium on Distributed Computer Control Systems (DCCS'98)*, IFAC (September 1998).

[Sha 1998]

L. Sha, J. B. Goodenough, & B. Pollak. "Simplex Architecture: Meeting the Challenges of Using COTS in High-Reliability Systems." *Crosstalk*, April 1998.

[Wallace 2005]

M. Wallace. "Modular Architectural Representation and Analysis of Fault Propagation and Transformation." *Electronic Notes in Theoretical Computer Science 14* (2005): 53-71.

6 Using the Vickrey-Clarke-Groves Auction Mechanism for Enhanced Bandwidth Allocation in Tactical Data Networks

Mark Klein, Daniel Plakosh, and Kurt Wallnau

6.1 PURPOSE OF THIS RESEARCH

We investigate the application of computational mechanism design to systems of interest to the U.S. Department of Defense (DoD), with particular emphasis on using computational mechanisms in highly dynamic, resource constrained, performance critical systems. To provide the investigation with clear and realistic scale dimensions, we developed an application framework that emulates a tactical data network, and investigated the use of one class of mechanism, the Vickrey-Clarke-Grove (VCG) auction, to efficiently allocate bandwidth on the tactical network to improve the quality of a common operating picture.

6.2 BACKGROUND

Systems make decisions. Control systems sense state and decide on control actions to keep key state parameters within a control envelope. Program trading systems monitor financial markets and decide when to buy and when to sell. Both use information obtained from the parts of the system to make these decisions. In many cases a decision maker can obtain the necessary information from the parts, and can make optimal decisions accordingly. However, this scheme can break down as systems get bigger. Two dimensions of scale are sufficient to demonstrate the point:

- The system is developed by and/or serves a growing *number of human users*; and human users have their *own incentives*. For example, in market trading systems and frequently encountered peer-to-peer systems, computational agents act on behalf of humans. In this setting the users must have an incentive to provide truthful information (e.g., how much a user values an item that is for sale) to the decision maker. Without this incentive, we can depend on users to hide or misrepresent this information, if it is in their interest to do so, even if this deception comes at the expense of the system as a whole.
- The system is increasingly distributed, and performs a growing *number and diversity* of tasks. For example, ad hoc sensor networks and network-centric combat systems will support a (possibly open-ended) number and variety of human tasks and computational agents. In these settings, it is impractical to assume that a decision maker can be constructed that knows enough about each of these tasks to impose an efficient solution. By analogy one can think of the economic distortions (e.g., supply, price, forecasting) introduced by centralized command economies. Such distortions become more prominent and severe as economies grow and become more diversified.

As systems scale up in these dimensions, interaction protocols are needed that are resistant to strategic manipulation by selfish users, and that efficiently aggregate information from the parts of a system to enable effective global decision making. Computational mechanism design is the discipline of designing such interaction protocols.

6.3 COMPUTATIONAL MECHANISM DESIGN

A *mechanism* is an institution such as an auction, voting protocol, or a market that defines the rules or protocols for how individuals are allowed to interact and governs the procedure for how collective decisions are made. *Mechanism design* is the sub-discipline of game theory and economics concerned with designing such institutions so that they achieve prescribed and desirable global outcomes. *Computational mechanism design*⁴ addresses situations where individuals are computational agents working on behalf of human agents.

Mechanism design has a deep research tradition in game theory, where it is sometimes known as implementation theory, and in microeconomics, where it is sometimes known as institution design. There are many examples of the practical use of mechanism design to achieve large-scale social objectives. McMillan offers a good discussion of the importance of getting the details of mechanism design right (in the U.S. public radio spectrum auction) and illustrates the consequences of mechanism defects (in the New Zealand radio spectrum auction) [Mas-Colell 1995].

Computational mechanism design has a more recent history of practical application. One substantial and well-documented use of computational mechanisms falls under the general heading of “e-commerce.” For example, it has been reported that over 98% of Google’s \$6.14 billion in revenue (as of 2006) is achieved through the use of an explicitly designed auction mechanism for allocating advertising space on web pages returned from keyword searches [Edelman 2007]. Another substantial application area in electronic commerce is in supply chain optimization [Staib 2001, Chen 2005, Sandholm 2006].

6.4 APPROACH

Our investigation focuses on the comparatively less-well-understood use of computational mechanisms to control or direct the behavior of large-scale, decentralized systems, and in particular to achieve an efficient allocation of computational resources using economic mechanisms. In this use, computational systems are viewed as virtual economies, with computational elements competing to use scarce computational resources to achieve their individual objectives.

The research literature provides examples of mechanisms being used to allocate processor cycles for scientific computing on the worldwide grid [Chen 2004]; for network routing [Holzman 2003]; for allocating network capacity [Anshelevich 2004, Anderson 2005]; for sensor fusion [Rogers 2006, Dang 2006]; for peer-to-peer systems [Chen 2004, Shneidman 2003]; for task allocation for autonomous robots [Gerkey 2002]; and for electricity markets [Hinz 2003]. This is not in any sense an exhaustive survey, and the use of market mechanisms to control complex system behavior is receiving considerable attention.⁵

Our investigation focuses on the use of economic mechanisms to achieve an efficient allocation of network bandwidth for a tactical data network. We developed a realistic emulation of a tactical

4 The term algorithmic mechanism design is also encountered in the literature.

5 See <http://www.marketbasedcontrol.com/> for example.

data network modeled on LINK-11, and developed a variant of a well-known auction mechanism to allocate network bandwidth for radar sensor fusion.

6.5 AUCTIONING BANDWIDTH ALLOCATION ON TACTICAL NETWORKS

LINK-11 is a collection of digital data link protocols for communications among a number of participating units. Communication on the link takes place by round robin, designated roll call. Each unit reports when requested to do so by a participating unit that has been designated as Net Control Station.

At 2250 BPS for data (a bit more for voice) network bandwidth is a scarce resource in LINK-11. Even its successor LINK-16 has only 28.8 KBS for data. To conserve bandwidth, LINK-11 uses a reporting responsibility (“R2”) protocol where exactly one platform assumes R2 for each radar contact, and only this platform reports data for that contact. While this approach has the virtue of conserving bandwidth, it sacrifices opportunities to fuse track data to improve the quality of the common operating picture.

Our concept is to auction additional quanta of bandwidth, and allow the participating units themselves to decide which track data will be most valuable. A computational auction mechanism automates this process.

Figure 6-1 shows a snapshot of a tactical network display developed for this study. In this snapshot, there are four participating units, each with its own region of observation highlighted. Track symbols displayed in white are tracks with only a single report, from the platform with reporting responsibility for that track. The contact is equally likely to be actually located at any position within the error ellipse of each track, depicted in red. Track symbols displayed in yellow are tracks that have been fused from at least two or more platforms, one of which has reporting responsibility. Note that the error ellipse for fused tracks is considerably reduced; this reflects the improved quality of the common operating picture yielded by the auction.

The amount of additional bandwidth allocated for fusion can be varied for each auction; the platforms themselves choose how that bandwidth should be allocated.

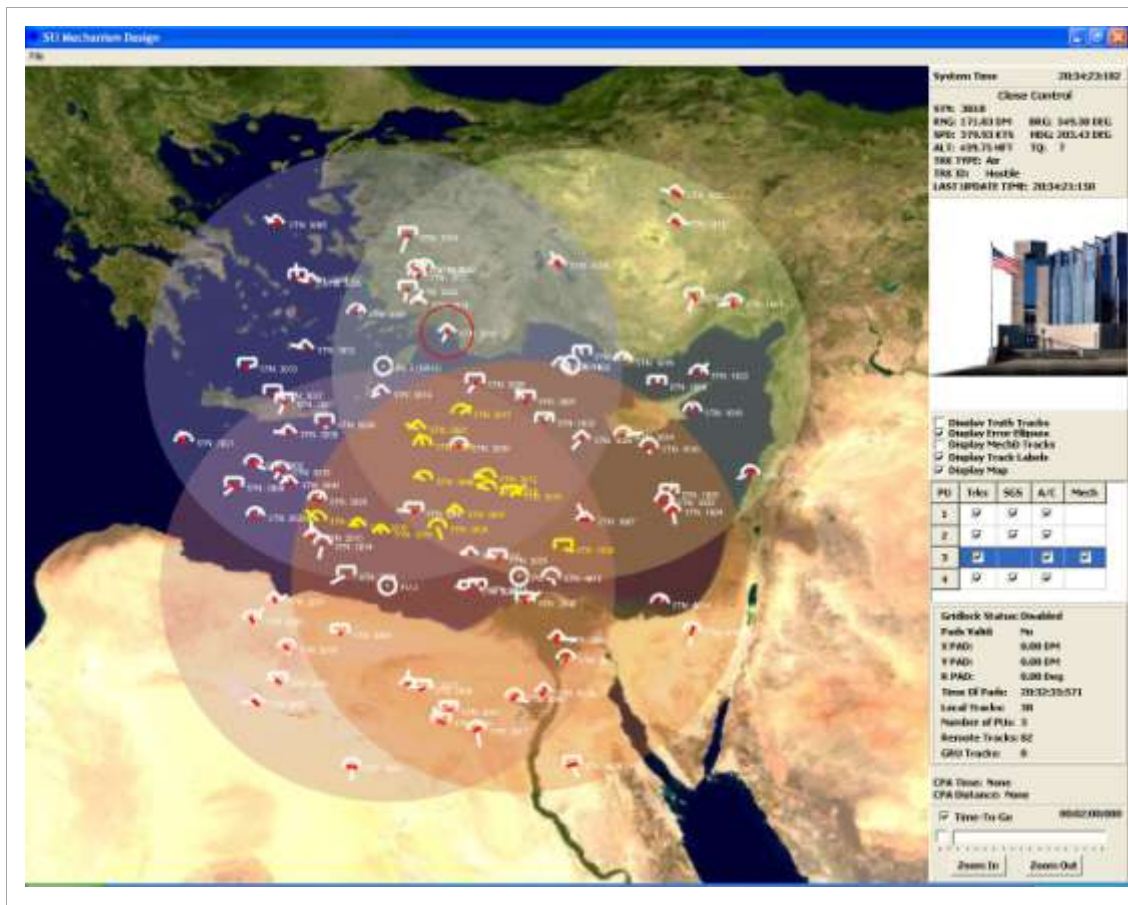


Figure 6-1: Tactical Display With Fused Track Data

6.5.1 Designing the Mechanism

Designing an auction mechanism induces several characteristic design questions. These questions, and our answers for LINK-11, are briefly summarized below.

Question 1: What is the scarce resource? We will auction additional quanta of bandwidth beyond the baseline R2 protocol, so that extra track data can be transmitted for data fusion. This fusion will improve the quality of the common operating picture.

Question 2: What is the criteria for desirable overall outcome. The mechanism must maximize the total information gain over all of the participants as a consequence of auctioning off spare bandwidth. Information gain is a quantifiable measure of the improvement in the quality common operating picture.

Question 3: What private information is possessed by participants that will determine their outcome preferences? Participants have private information about which tracks they can see, and about the quality of their track data. Participants will misrepresent their private information to the auctioneer if doing so will induce an outcome that is more favorable than if they tell the truth.

Question 4: How are participant preferences represented in a payoff structure? Each participant is driven by self interest. Self-interest stems from this hypothesized, but plausible, doctrine:

- Survivability of the individual participant depends on the survivability of the battle group, which in turn depends on maximizing information gain of the whole group.
- “After action reviews,” which lead to promotions and other rewards, use marginal contribution to total information gain as an important evaluation criterion.

This incentivizes every participant to maximize their contribution to the group's information gain rather than increasing one's own information gain.

Eq.1

$$u_i(Z, F^*) = v_i(Z, F^*) - \left[\sum_{j \neq i} v_j(Z, F_{-i}^*) - \sum_{j \neq i} v_j(Z, F^*) \right]$$

Participant i's payoff
Participant i's information gain
Participant i's payment

The incentivized payoff structure for the bandwidth auction is defined in Eq. 1, which reflects each participant's marginal contribution to total information gain, where u_i and v_i are payoff and value functions for participant i , respectively; Z is the information that participant i has for all tracks; F^* and F_{-i}^* is the optimal bandwidth allocation with and without participant i included in the auction, respectively.

The above doctrine incentivizes each participant to maximize its payoff; and, its payoff is maximized by maximizing the information gain of the whole group.

Question 5: What are the rules of the auction? An auction defines the bidding rules, the resource allocation approach, and the “payments” made by each participant for the resources they receive. We used the Vickrey-Clarke-Groves (VCG) mechanism as our starting point. The VCG auction is a generalization of the second price sealed bid auction. The VCG auction has the key property of “incentive compatibility,” which ensures that each participating unit will maximize their payoff only by truthfully revealing their private information.

The result is that the needs of the individual are aligned with the needs of the many. This is the “trick” of mechanism design.

6.6 ASSESSING THE AUCTION

Figure 6-2 depicts a snapshot of the portion of the “Net Control Station” interface used to study the auction at runtime.

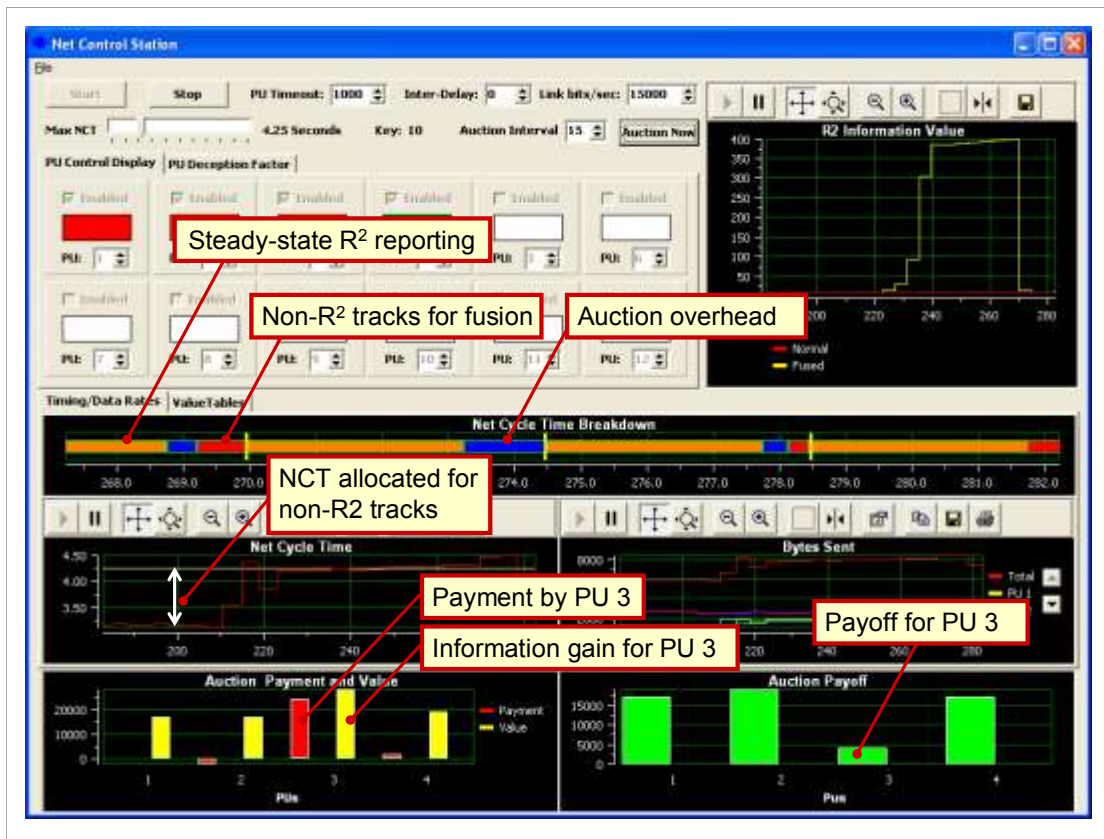


Figure 6-2: Studying the Runtime Effects of the Auction

The vertical, two-headed arrow labeled “NCT allocated...” shows the quantum of bandwidth auctioned for the purpose of data fusion. The horizontal bar labeled “Steady state R2 reporting” shows how bandwidth is used, and includes the cost of running the auction itself. The auction is run periodically, for example once every fifteen network cycles.

The economic outcome for one auction is shown at the bottom of Figure 6-2. In this example, participating unit 3 (PU 3) gains the most information but also makes the largest payment. PU 3’s payment represents its adverse impact on the other participants. That is, if PU 3 were not in the auction, its payment (red bar) would be distributed as information gain (yellow bar) among the remaining participants.

Improvement in the quality of the common operating picture as a result of this auction is shown by the yellow “fused” tracks displayed in Figure 6-1. The error ellipse for these tracks (shown in red on each track) has been substantially reduced (that is, their accuracy has been increased) as a result of the auction.

6.7 CONTRIBUTION OF THIS WORK

The viability of techniques such as computational mechanism design can be established only when they are applied to problems of sufficient scale and complexity to expose the practical limitations of the techniques in question. When theory is applied to practice, practice invariably

“pushes back.” It is often the case that this “push back” leads to identification of opportunities to advance and refine the underlying theory of a technique that would never have been identified but for the confounding, and impossible-to-predict, effects of real-world problems. Our work builds on earlier work by Dash et al. [Rogers 2006, Dang 2006], but adds significantly to the complexity (and, we claim, the resulting fidelity) to the experimental setting. In addition, our work emphasizes the importance of accounting for human incentives in the process of designing computational mechanisms. With this philosophical background, our work has made three key contributions.

1. We developed an application framework⁶ that exhibits sufficient scale and dynamic complexity to study the feasibility of computational mechanism design in a practical setting. The application framework emulates a combat tactical data network, and includes much of what is required to construct a common operating picture from radar sensor data.
2. We designed and implemented a variant of the well-known Vickrey-Clarke-Grove auction for use in the application framework. The auction is used to efficiently allocate a fixed, but selectable, amount of network bandwidth to permit fusion of additional sensor data to improve the common operating picture. One novel aspect of our auction is that participants are both buyers and sellers of information, and each can obtain value from buying and selling.
3. Finally, and perhaps most importantly, we demonstrated that computational mechanisms can be used to implement distributed, value-based resource allocation schemes in the kind of highly dynamic, resource constrained, performance critical systems found in DoD combat systems. Such systems are *non plus ultra* for evaluating the practicality for using computational mechanisms to control the behavior of complex systems.

6.8 CONCLUSIONS

Our research provides strong evidence that:

- Computational mechanism *design* provides new and useful design principles for the design of complex systems, especially those that support users who may have distinct incentives.
- Computational *mechanisms* can be used in performance critical, highly dynamic settings such as those found in tactical data networks, with behavior that is predictable using strong underlying game- and microeconomic theory.

These results, we believe, are applicable to a much broader class of system than tactical data networks; in fact, the research is primarily intended to study mechanism design and only secondarily to study its use in a particular setting.

Nonetheless, the research also demonstrates that the well-known VCG auction can be useful in existing DoD tactical data networks as a tool for providing incremental improvements in the quality of a common operating picture. In addition, we have identified avenues for further refining the VCG auction, and for using market mechanisms, to embrace different tactical data network settings.

Last, and concretely, we have provided the research community with a robust application framework for studying computational mechanisms. This sort of framework—and others like it—will

⁶ The implementation has been packaged for use by external research collaborators and has already been made available to select researchers at Naval Postgraduate School and Harvard University.

be enormously useful to close the gap between the sometimes alien research traditions of game theory and microeconomics and the practical requirements of software and systems engineering of complex systems.

6.9 REFERENCES

[Anderson 2005]

Edward Anderson, Frank Kelly, & Richard Steinberg. “A Contract and Balancing Mechanism for Sharing Capacity in a Communication Network.” Computing and Markets, Dagstuhl Seminar Proceedings. Schloss Dagstuhl, Germany, March-July 2005. Lehman, Muller & Sandholm (eds). Internationales Begegnungs- und Forschungszentrum fuer Informatik (IBFI), 2005.

[Anshelevich 2004]

Elliot Anshelevich, Anirban Dasgupta, Jon Kleinberg, Eva Tardos, Tom Wexler, & Tim Roughgarden. “The Price of Stability for Network Design with Fair Cost Allocation.” In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS'04)*, IEEE Computer Society, 2004.

[Chen 2004]

Ming Chen, Guangwen Yang, & Xuezheng Liu. “Gridmarket: A Practical, Efficient Market Balancing Resource for Grid and P2P Computing,” 612-619. *Grid and Cooperative Computing: Second International Workshop, GCC 2003. Lecture Notes in Computer Science*, Volume 3033. Shanghai, China, December 2003. Springer, 2004.

[Chen 2005]

Rachel Chen, Obin Roundy, Rachel Zhang, & Ganesh Janakiraman. “Efficient Auction Mechanisms for Supply Chain Procurement.” *Source Management Science* 51, 3 (March 2005): 467-482.

[Dang 2006]

V. D. Dang, R. K. Dash, A. Rogers, & N. R. Jennings. “Overlapping Coalition Formation for Efficient Data Fusion in Multi-Sensor Networks.” In *Proceedings of the National Conference on Artificial Intelligence*, Vol 21; Part 1 (2006): 635-640.

[Edelman 2007]

Benjamin Edelman, Michael Ostrovsky, & Michael Schwarz. “Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords.” *Journal of American Economic Review* (2007).

[Gerkey 2002]

Brian P. Gerkey & Maja J. Mataric. “Sold! Auction Methods for Multirobot Coordination.” *IEEE Transactions on Robotics and Automation* 18, 5 (October, 2002).

[Hinz 2003]

Juri Hinz. “Optimal Bid Strategies for Electricity Auctions.” *Journal of Mathematical Methods of Operations Research*, Physica Verlag, 57, 1 (April 2003).

[Holzman 2003]

Ron Holzman & Nissan Law-Yone. “Network Structure and Strong Equilibrium in Route Selection Games.” *Journal of Mathematical Social Sciences* 46, 2 (October 2003): 193-205.

[Mas-Colell 1995]

Andreu Mas-Colell, Michael Dennis Whinston, & Jerry R. Green. *Microeconomic Theory*. New York: Oxford University Press, (ISBN: 0195073401).

<http://www.loc.gov/catdir/enhancements/fy0604/95018128-d.html>

[Rogers 2006]

A. Rogers, R. K. Dash, N. R. Jennings, S. Reece, & S. Roberts. “Computational Mechanism Design for Information Fusion within Sensor Networks.” 9th International Conference on Information Fusion, July 2006.

[Sandholm 2006]

T. Sandholm. “Expressive Commerce and Its Application to Sourcing.” In *Proceedings of the Eighteenth Conference on Innovative Applications of Artificial Intelligence (IAAI06)*, June 2006. Menlo Park, CA.

[Schneidman 2003]

Jeffrey Shneidman & David C. Parkes. “Rationality and Self-Interest in *Peer to Peer Networks*.” In *Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS'03)*, 2003.

[Staib 2001]

Wing Commander Margaret Staib. “Sustainment Procurement in the Air Force – U.S. Department of Defense.” *Air Force Journal of Logistics*, Summer, 2001.

REPORT DOCUMENTATION PAGE			<i>Form Approved</i> <i>OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave Blank)		2. REPORT DATE July 2008		3. REPORT TYPE AND DATES COVERED Final
4. TITLE AND SUBTITLE Results of SEI Independent Research and Development Projects			5. FUNDING NUMBERS FA8721-05-C-0003	
6. AUTHOR(s) Len Bass, Dionisio de Niz, Jörgen Hansson, John Hudak, Peter H. Feiler, Don Firesmith, Mark Klein, Kostas Kontogiannis, Grace A. Lewis, Marin Litoiu, Daniel Plakosh, Stefan Schuster, Lui Sha, Dennis B. Smith, & Kurt Wallnau				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2008-TR-017	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116			10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2008-017	
11. SUPPLEMENTARY NOTES				
12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS			12B DISTRIBUTION CODE	
The Software Engineering Institute (SEI) annually undertakes several independent research and development (IRAD) projects. These projects serve to (1) support feasibility studies investigating whether further work by the SEI would be of potential benefit and (2) support further exploratory work to determine whether there is sufficient value in eventually funding the feasibility study work as an SEI initiative. Projects are chosen based on their potential to mature and/or transition software engineering practices, develop information that will help in deciding whether further work is worth funding, and set new directions for SEI work. This report describes the IRAD projects that were conducted during fiscal year 2007 (October 2006 through September 2007).				
14. SUBJECT TERMS independent research and development, IRAD, architectural design decisions, software architecture, software-intensive systems, real-time systems, embedded systems, fault tolerance, fault management, fault containment, Vickrey-Clarke-Groves, auction mechanism, VCG, computational mechanism design			15. NUMBER OF PAGES 53	
16. PRICE CODE				
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	