Final Report

Making Production Operating System Kernels Adaptive: Incremental Specialization in Practice (Synthetix Project)

Oct. 97

Abstract

We summarize the results produced by the DARPA contract "Making Production Operating System Kernels Adaptive: Incremental Specialization in Practice", also known as the Synthetix Project. The main objective of the project is to develop specialization technology to improve the modularity, adaptiveness, and performance of production operating system code. The main results are software toolkits that help programmers to build systems with specialized components that maintain their modularity and portability. The project is being extended by the Microlanguage and Immunix projects. We have released the Synthetix Specialization Toolkit and Software Feedback Toolkit.

DARPA Order Number:	B752
Name of contractor:	Oregon Graduate Institute of Science & Technology
	P.O. Box 91000
	Portland, OR 97291-1000
Contract Number:	N00014-94-1-0845
Principal Investigator:	*Calton Pu*
	Phone: (503) 690-1214
	FAX: (503) 690-1553
	Email: calton@cse.ogi.edu
	URL: http://www.cse.ogi.edu/~calton
Project Title:	Synthetix Project

Sponsored by
Defense Advanced Research Projects Agency, CSTO/ITO
DARPA Order No. B752
Issued by ONR under grant N00014-94-1-0845

Contents

1 Task Objectives **1**
- 1.1 Summary of Project Objectives . . . 1
- 1.2 Synthetix Deliverables . . . 1

2 Technical Problems **2**

3 General Approach and Methodology **3**
- 3.1 Methodical Specialization . . . 3
- 3.2 Toolkit Based Specialization . . . 4
- 3.3 Interaction With Other OGI Projects . . . 5

4 Technical Results **6**
- 4.1 Papers and Publications . . . 6
- 4.2 Presentations and Interactions . . . 9
- 4.3 Software Deliverables . . . 11
  - 4.3.1 Summary of Deliverables . . . 11
  - 4.3.2 June–Dec 1994 . . . 12
  - 4.3.3 Jan–Dec 1995 . . . 12
  - 4.3.4 Jan–Dec 1996 . . . 12
  - 4.3.5 Jan–June 1997 . . . 13

5 Important Findings **14**
- 5.1 Early Findings . . . 14
- 5.2 Summary of Project Findings . . . 15

6 Significant Hardware Developments **15**

7 Special Comments on Industrial Collaboration **15**

8 Implication for Further Research **16**

Bibliography . . . 17

1 Task Objectives

1.1 Summary of Project Objectives

This is the final report for the Synthetix project grant (DARPA/ONR grant N00014-94-1-0845). The Synthetix project is also strengthened and complemented by the Microlanguage contract (number F19528-95-C-0193), as well as augmented with another DARPA grant in the Information Survivability program (Immunix, DARPA grant F30602-96-1-0331). We will report the progress and achievements of the project as a whole, distinguishing the results from each contract funding as appropriate.

For the Synthetix grant, the concrete deliverables fall into three categories: a methodology for applying it to existing operating system kernels, tools to support incremental specialization, and kernel code that demonstrates its use in practice. Similarly, for the Microlanguage contract, the concrete deliverables fall into three groups: techniques, microlanguages plus their associated software tools, and experimental systems that use microlanguages.

Synthetix specialization is primarily targeted towards situations where the systems software can *infer* the kind of specialization to apply. The goal of microlanguages is to expand the power and applicability of specialization techniques and toolkit to situations where systems programmers and application programmers may *direct* the specialization process. Some of the basic specialization techniques and tools apply equally to inferred and directed specialization, but many methods and tools need to be refined and redesigned to take into account both kinds of specialization.

1.2 Synthetix Deliverables

We will develop a methodology for the design and implementation of operating systems (and their components) that use incremental specialization. Concretely, this methodology will outline a series of steps that will guide kernel developers both in the development of new kernels from scratch, and in the modification of existing kernel code. The methodology will apply to existing monolithic kernels and micro-kernels.

We will develop a toolkit that includes a C preprocessor and compiler with support for: (a) fine-grain modularity, (b) the specification of invariants and guards, and (c) the generation of templates that can be instantiated post compile time. In addition, the toolkit will also include a run-time kernel supporting dynamic code generation, dynamic linking, and possibly some code optimization such as constant folding and loop unrolling.

Another important part of the Synthetix specialization toolkit is the support for fine-grain adaptation using software feedback. The software feedback toolkit will include (1) a software implementation of generic filters to be used in the kernels, (2) software tools for filter design, development, and testing, (3) composition tools to combine elementary filters into more sophisticated filters, and (4) guard programs that detect input oscillation beyond

the specified filter range.

The specialization toolkit is demonstrated in production operating system kernels. Our initial experiments were conducted on HP-UX, the HP commercial version of Unix, particularly in the Unix File System component. Experiments have been conducted on SUNOS (the RPC component) and Linux, a public domain Unix operating system (the signal handler component).

For brevity of presentation and focus on Synthetix we omit the list of deliverables from the Microlanguages contract and the Immunix grant.

2 Technical Problems

Modern operating systems have been growing in size and complexity due to the constant pressure for additional functionality. As the variety of applications widens, and hardware platforms become increasingly powerful, the operating system has been used for different purposes. Consequently, operating system code has been stretched far beyond its original intent. For example, file systems optimized for block size access typically do not support byte-by-byte access efficiently. Although libraries such as `stdio` keep work outside the kernel, the result is the growing size and complexity of both the kernel code and the systems software related to the kernel.

To aggravate the problem, kernel operations are encapsulated within blackbox boundaries, so users cannot customize the kernel very easily. Specialization of the operating system kernel has been developing as a promising technique in a number of projects such as Synthesis [25, 19], x-kernel [21], Spin [2], and Synthetix [9, 22]. However, specialization has encountered two limitations in its wide application. First, it has been applied successfully in specific domains, but each domain seems different enough to require a large new effort. Second, efforts to make customization easy, such as Spin, have yet to address the concerns of quality control, interference with other kernel modules, maintainability, and system evolution.

Specialization is a well known technique that has been applied in an *ad hoc* way in operating system kernels (and all the other layers of systems and application software). The Synthetix approach aims for a methodical approach to specialization and the construction of tools to help kernel programmers apply specialization techniques to systems software. In making specialization a methodical approach, we have identified three main technical problems:

- The identification and explicit specification of assumptions (called *quasi-invariants*) made by the specialized code. Any quasi-invariants that remain implicit may cause bugs in the other components of the system that call the specialized code, since *all* of the quasi-invariants must be true for the specialized code to function correctly.

- The guarding of the quasi-invariants during execution of the specialized code. After the quasi-invariants have been made explicit, the system must guard them during the execution. This is particularly difficult to handle in kernels, since events may render quasi-invariants false during the execution of specialized kernel calls.

- Specialization can be seen as adaptation at a coarse granularity, when quasi-invariants become violated. An important concern for adaptive software is a lower cost mechanism for fine-grain adaptation. Most adaptive systems, even those based explicitly on feedback mechanisms, are *ad hoc* with hand customization. The development of a general purpose software feedback toolkit useful for many applications remains a serious technical problem.

In addition to these technical problems, our industry partners made it very clear to us that a fundamental requirement for technology transfer is the availability of tools for their programmers. Therefore, our solutions must be packaged in tools that help programmers adopt these solutions. We consider this fourth problem also an engineering problem, in addition to the three technical problems, since the development of tools influenced the development of the methodology and the technology itself. In particular, most of the other projects developing techniques for the enhancement of *ad hoc* specialization have missed this important consideration.

3 General Approach and Methodology

3.1 Methodical Specialization

The first part of the Synthetix approach to the technical problems listed in Section 2 is the design and refinement of a methodical approach to specialization of systems software. In constrast to *ad hoc* specialization, we enumerate the steps through which a programmer may develop specialized code, whether by hand or by using tools. By dividing the steps carefully, we are able to design specialization tools to help the programmer during these steps. We divide the specialization process into 5 steps.

The first step is the analysis of the general algorithm and the specialized versions. We assume that each specialized piece of code has a general algorithm behind it. Although the general algorithm does not have to be necessarily implemented in the system, we avoid considering the specialized code in isolation. By comparing the different versions of specialized code with the general algorithm, the goal is to make as many of the quasi-invariants explicit as feasible. In the specialization of the Unix File System example [22], examples of quasi-invariants include: (1) exclusive access to the file and (2) sequential access of the file.

The second step, once the quasi-invariants have been identified, is to specify them and identify the other system components that interact with these quasi-invariants. In the Unix File System example, the exclusive access quasi-invariant interacts with some obvious

modules (e.g., open of the file) and some non-obvious ones (e.g., sending the file descriptor in a message that allows the receiver to access the same file). It is important to identify *all* of the interactions, since missing any constitutes a bug in the overall system.

The third step is the design and implementation of the specialized code and all the identified guards for the quasi-invariants. The specialized code takes full advantage of the quasi-invariants to simplify the code with respect to the general algorithm. At the same time, the guards watch over the possible violations of the quasi-invariants to make sure the specialized code executes only during the moments the quasi-invariants are actually true. The first three steps form an interactive process, where the design of the specialized code and guards influence the choice of which quasi-invariants are used for specialization.

The fourth step is the refinement of specialized code and guards at a micro level. Even when we preserve the quasi-invariants, there are many implementation choices for each piece of specialized code and its guards. This is particularly the case when the specialized code is evolving (for a number of reasons), and the specialized code and guards must be regenerated after each change.

The fifth step in the methodology is the evaluation of the specialized code and guards through measurements. During and after implementation, it is important to evaluate their performance and robustness through actual measurements. Further tuning may result in additional design and implementation changes. This final step forms the final link in the interactive development of the entire specialization process.

3.2 Toolkit Based Specialization

We have built the Synthetix Specialization Toolkit (Section 4.3.2) to assist kernel programmers in the specialization process. It consists of two guarding programs (TypeGuard and MemGuard), a concurrent replugger, and access to the Tempo-C program specializer. The toolkit is available through the World Wide Web at the URL:

`http://www.cse.ogi.edu/DISC/projects/synthetix/toolkit/`

An example of specialization using the toolkit is the specialization of Linux signal handling, which is described in a tutorial in the web page. In general, tool-based specialization follows the same methodology described in Section 3.1. In the specialization of Linux signal handling, TypeGuard is used in step 2 in the identification of possible violations of quasi-invariants. Tempo-C is used in step 3 and 4 for the generation and optimization of signal code. MemGuard is used in the development and debugging of the specialized kernel. The replugger is not used directly in this example due to the nature of typical signal handling, but it is used in some other experiments.

3.3 Interaction With Other OGI Projects

The Synthetix project has been the leading component of a set of DARPA supported projects at OGI, summarized below (all of them have Calton Pu as the lead PI).

- Synthetix project (94–97): described in this report.
- Microlanguages project (95–98): design and implementation of microlanguages for directed specialization, in contrast to the inferred specialization done in Synthetix.
- Immunix project (96–99): application and extension of specialization techniques and the Synthetix Specialization Toolkit to increase the information survivability of operating systems.
- Quasar Microfeedback project (97–2000): investigation of feedback system properties within and outside the assumption specifications, and the construction of a composable microfeedback toolkit.

We briefly summarize the influence of Synthetix on these projects and the information flow as well as technology adoption and transfer among them.

Microlanguages The main objective is to apply discipline to specialization through *microlanguages*, meta-interface languages that characterize how an interface is used, and tailored for each application. Each microlanguage specifies all the specialized execution paths in an appropriate systems software component, e.g., file system, network protocol, and a client/server interface for an application area. On the client's side, a sufficiently rich microlanguage provides a high level declarative way to express customization, rather than by writing code in a (third-generation) system implementation language that then has to be "sanitized" and inserted into the kernel. On the systems side, kernel designers determine the scope of specialization during the design and implementation of microlanguages. The Synthetix Specialization Toolkit is being used in the Microlanguages project, which extends it for use by directed specialization.

Immunix The main objective is to increase OS security fault resistance through systematic specialization. Synthetix specialization of operating system kernels uses an explicit specification of *invariants* that allow dynamic generation of code at run-time to improve operating system performance while preserving modularity. All the specialized cases preserve the same OS kernel functional interface semantics. Reusing these performance-oriented invariants and introducing *artificial* invariants where necessary, we can generate operating system modules as complex and varied as we specify them to be, while maintaining the operating system functionality. The Specialization Toolkit has helped directly (e.g., the construction of MemGuard/StackGuard against stack overflow attacks) and indirectly (e.g., the classification of responses to attacks into a two-dimensional table in terms of interface restriction and permutation, and implementation restriction and permutation).

Quasar Microfeedback The main objective is to build a toolkit of software feedback coomponents with four functional properties:

1. (1) Working characteristics: how quickly the microfeedback responds to changes in input, and what is the maximum error (delay) in its response;
2. (2) boundaries of applicability: under which conditions the microfeedback fails to respond appropriately, and how badly it fails;
3. (3) feedback interaction: how aggressive, competitive, and cooperative the microfeedback is in the presence of other feedback mechanisms;
4. (4) meta adaptation: any changes on the above three properties due to adjustment of microfeedback parameters or its replacement when crossing its applicability boundaries.

The Synthetix Software Feedback toolkit forms the starting point of the Microfeedback Toolkit. We are analyzing the properties of feedback components and building the support for composing them.

4 Technical Results

The main technical results from the Synthetix project consist of publications of papers and software toolkits. We have also had significant collaboration and interaction with the international research community. In Section 4.1 we outline a selection of our publications and relevant papers written during this period. Section 4.2 summarizes the external presentations and interactions with other groups. In Section 4.3 we summarize the Synthetix contract deliverables, including the software published up to now (September 97).

4.1 Papers and Publications

During the June–Dec 1994 period:

1. Calton Pu presented the main ideas of the Synthetix project [26] at the 1994 OOPSLA Workshop on Flexibility in Systems Software.

During the Jan–Dec 1995 period:

1. Shanwei Cen presented a demonstration [4] of our real-time distributed MPEG video/audio player at the 1995 ACM International Conference on Multimedia. The demo showcased our software-feedback based software, which has been freely available on the Web since April 1995. The player was described in a separate paper [5] in the 1995 International Workshop on Network and Operating System Support for Digital Audio and Video.

2. Our paper [13] on "Adaptive Methods for Distributed Video Presentation" has appeared in a special Symposium on Multimedia of the ACM Computing Surveys journal. The paper analyzes the problems and solutions for delivering real-time multimedia presentations across the Internet, with special attention to our approach based on software feedback and our demonstration software.

3. Jonathan Walpole presented our paper [22] at the 1995 ACM Symposium on Operating Systems Principles (SOSP). The paper described our experience with applying specialization techniques systematically to a commercial operating system (HP-UX). In this paper, we have demonstrated the potential gains of specialization in performance and modularity on the one hand, and on the other hand, the difficulties with verifying the correctness of the specialized code, requiring powerful software tools for the large scale application of specialization. The SOSP is the bi-annual premier conference in operating systems research.

During the Jan–Dec 1996 period:

1. Our paper [8] on "A General Approach for Run-Time Specialization and its Application to C" has been presented in the 1996 ACM Symposium on Principles on Programming Languages (POPL). This paper describes the theoretical foundation of the C on-line partial evaluator Consel is developing in France as part of the Synthetix specialization toolkit. The POPL is the annual premier conference on programming languages research.

2. Our paper [31] on "Safe Operating System Specialization: the RPC Case Study" has been presented at the First Workshop on Compiler Support for Systems Software (WCSSS), Arizona, February 1996. This paper describes the experiments in applying specialization to the RPC code of the Chorus microkernel.

3. Our paper [7] on "A Uniform Approach to Compile-Time and Run-Time Specialization" has been presented at the 1996 Dagstuhl Workshop on Partial Evaluation, Germany, February 1996. This paper describes the general approach of the Rennes group to specialization at both compile-time and run-time.

4. The paper [14] on "A Wait-free Algorithm for Optimistic Programming: HOPE Realized", by Crispin Cowan and Hanan L. Lutfiyya, was presented at the 1996 International Conference on Distributed Computing Systems, Hong Kong, May 1996.

5. Our paper [30] on "A Uniform Automatic Approach to Copy Elimination in System Extensions via Program Specialization", by Volanschi, E.-N., Muller, G., Consel, C., Hornof, L., Noye, J. and Pu, C., was published as IRISA Research Report No. 1021, Rennes, France, June 1996. It was submitted to the 1996 Symposium on Operating System Design and Implementation, Seattle, October 1996.

6. Our paper [10] "Automated Guarding Tools for Adaptive Operating Systems", by Crispin Cowan, Andrew Black, Charles Krasic, Calton Pu, and Jonathan Walpole was submitted to the 1996 Symposium on Operating System Design and Implementation, Seattle, October 1996.

7. Our paper [29] "Managing Adaptive Presentation Executions in Distributed Multimedia Database Systems" by H. Thimm, W. Klas, J. Walpole, C. Pu, and C. Cowan, presented in the 1996 IEEE International Workshop on Multimedia Database Management Systems, Blue Mountain Lake, New York, August 1996.

8. Our paper [3], "System Support for Mobility", by A. Black and J. Inouye, was presented at the ACM SIGOPS European Workshop in Ireland, September 1996.

9. Our paper [12] "Specialization Classes: An Object Framework for Specialization", by Crispin Cowan, Andrew Black, Charles Krasic, Calton Pu, and Jonathan Walpole of OGI, and Charles Consel and Eugen-Nicolae Volanschi of University of Rennes / IRISA, accepted for presentation in the 1996 International Workshop on Object Orientation in Operating Systems, Seattle, October 1996.

10. Our paper [23] "A Specialization Toolkit to Increase the Diversity in Operating Systems" by C. Pu, C. Cowan, A. Black and J. Walpole of OGI, and C. Consel of IRISA/University of Rennes, accepted for presentation in the ICMAS Workshop on Immunity-Based Systems, Kyoto, Japan, December 1996.

11. Our paper [24] "Microlanguages for Operating System Specialization" by C. Pu, C. Cowan, A. Black and J. Walpole of OGI, and C. Consel of IRISA/University of Rennes, accepted for presentation in the POPL Workshop on Domain-Specific Languages, Paris, January 1997.

12. Crispin Cowan presented the specialization class paper [12] at the International Workshop on Object-Oriented in Operating Systems, Seattle, October, 1996.

13. Veronica Baiceanu [1] and Dan Revel [28] presented their work on CPU and disk scheduling for multimedia applications at the Workshop on Resource Allocation Problems in Multimedia Systems, Washington, D.C., December 1996.

14. Calton Pu presented our ideas for improving operating system survivability [23] at the Immunity-Based Systems Workshop, Japan, December 1996.

During the Jan–June 1997 period:

1. Andrew Black presented the paper [24] "Microlanguages for Operating System Specialization", by Calton Pu, Andrew Black, Crispin Cowan, Jonathan Walpole (OGI), and Charles Consel (Univ. of Rennes/IRISA), at the ACM SIGPLAN Workshop on Domain-Specific Languages January 18, 1997 Paris, France (in association with POPL'97).

2. Crispin Cowan presented the paper "Immunix: Survivability through Specialization", by Crispin Cowan and Calton Pu, at the Information Survivability Workshop, organized by SEI, San Diego, 2/12.

3. Jonathan Walpole and others wrote the paper [32] "A Toolkit for Specializing Production Operating System Code", submitted to the ACM SIGOPS Symposium on Operating Systems Principles (SOSP'97).

4. Gilles Muller, Renaud Marlet, Eugen-Nicolae Volanschi, Charles Consel, Calton Pu, and Ashvin Goel wrote the paper [20] "Fast, Optimized SUN RPC Using Automatic Program Specialization", submitted to the ACM SIGOPS Symposium on Operating Systems Principles (SOSP'97).

5. Dan Revel, Crispin Cowan, Dylan McNamee, Calton Pu and Jonathan Walpole wrote the paper [27] "Predictable File Access Latency for Multimedia", submitted to IFIP Fifth International Workshop on Quality of Service (IWQOS'97).

6. Jon Inouye, Jonathan Walpole, and Jim Binkley, wrote the paper [16] "Physical Media Independence: System Support for Dynamically Available Network Interfaces", Technical Report CSE-97-001, Oregon Graduate Institute, January 1997.

7. Jon Inouye, Shanwei Cen, Calton Pu and Jonathan Walpole, wrote the paper [15] "System Support for Mobile Multimedia Applications", appeared in the Proceedings of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97), St. Louis, Missouri, May 19-21, 1997.

4.2 Presentations and Interactions

During the June–Dec 1994 period:

1. On 6/03/94, Calton Pu presented the talk entitled "Update on The Synthetix Operating System Project" at Hewlett-Packard, Cupertino, California, hosted by Mr. John Sontag.

2. On 8/29/94, Calton Pu presented the talk entitled "The Synthetix Operating System Project" at Honeywell Technology Center, Minneapolis, hosted by Dr. Steve Vestal.

3. On 9/28/94, Calton Pu presented the talk entitled "The Synthetix Operating System Project" at Georgia Institute of Technology, Atlanta, hosted by Prof. Mustaq Mahamad.

4. On 9/29/94, Calton Pu presented the talk entitled "The Synthetix Operating System Project" at Clark-Atlanta University, Atlanta, hosted by Prof. Alvin Lim and Prof. Karsten Schwan.

During the Jan–Dec 1995 period:

1. On 04/19/95, Calton Pu presented the talk entitled "Kernel Specialization Tools and Techniques: Synthetix Project", at Technical University of Darmstadt, Darmstadt, Germany, hosted by Dr. Prof. Alejandro Buchmann.

2. On 04/24/95, Calton Pu presented the talk entitled "Kernel Specialization Tools and Techniques: Synthetix Project", at University of Rennes, Rennes, France, hosted by Prof. Charles Consel.

3. On 08/17/95, Calton Pu presented the talk entitled "Kernel Specialization Tools and Techniques: Synthetix Project", at DEC CEC, Karlsruhe, Germany, hosted by Dr. Lutz Heuser.

4. On 08/18/95, Calton Pu presented the talk entitled "Kernel Specialization Tools and Techniques: Synthetix Project", at University of Kaiserslautern, Germany, hosted by Prof. Jurgen Nehmer.

5. On 11/09/95, Calton Pu presented the talk entitled "Kernel Specialization Tools and Techniques: Synthetix Project", at Xerox PARC, California, hosted by Gregor Kiczales.

During the Jan–Dec 1996 period:

1. Andrew Black presented the overview of the project at the University of Glasgow in a visit during March 1996.

2. Adaptive Operating Systems (overview of Synthetix in the large) presented at the University of Waterloo department of Computer Science on June 20th by C. Cowan.

3. Adaptive Real-time Scheduling for Multimedia Quality of Service in a UNIX Environment. Guest lecture to the University of Waterloo department of Computer Science CS 452 (Real Time Systems) course. Presented on June 21st by C. Cowan.

4. Andrew Black presented the paper [3] at the European SIGOPS Workshop.

5. Heiko Thimm, our collaborator from GMD/IPSI, who was visiting OGI for 4 months during Summer, presented the paper [29] at the Multimedia Database Workshop, August 1996.

6. Jonathan Walpole presented the talk on "Quality of Service Specification and Adaptive Resource Management for Distributed Multimedia Systems" for the Intel Research Council and Research Relations at Hillsboro, Oregon, September 1996.

7. Calton Pu presented a talk on "Overview of the Synthetix Project" at IRISA/University of Rennes, France, September 1996.

8. Jonathan Walpole gave the talk on Quality of Service Specification and Adaptive Resource Management for Large Distributed Systems at the Quorum PI meeting (Dallas, November 1996).

9. Crispin Cowan visited CMU, Prof. Daniel Jackson, on December 9th, 1996, to discuss collaboration. We are actively studying the use of `lackwit`, their software analysis tool, for the static verification of guarded code. The OGI group has sent real code to be analyzed by the CMU group as the preliminary evaluation of the tool robustness for production code.

During the Jan–June 1997 period:

1. Andrew Black visited University of Glasgow (Glasgow, United Kingdom, 1/24), and gave a talk on "Microlanguages for Operating System Specialization".

2. Charles Consel visited Xerox PARC (Palo Alto, California, 2/21), Microsoft (Redmonds, Washington, 2/24), University of Washington (Seattle, Washington, 2/26), and gave talks on the Tempo Specializer and system support for microlanguages.

3. Crispin Cowan visited Intel's Data Security Group (Hillsboro, Oregon, 3/12) to give a talk on "Immunix: Survivability through Specialization".

4. Calton Pu was an invited panelist in the panel on "Survivability in the Face of Malicious Attack", in the Sixth IFIP Working Conference on Dependable Computing for Critical Applications (DCCA-6), panel chaired by Teresa Lunt of DARPA/ITO.

5. Jonathan Walpole gave the presentation "The Design of a Multimedia Player with Advanced QoS Control" at the Advanced Information Processing and Analysis Symposium, Washington, DC, 3/26.

4.3 Software Deliverables

4.3.1 Summary of Deliverables

Synthetix deliverables consist of a systematic methodology for applying specialization to operating systems code, and toolkits for specialization as well as software feedback adaptation. The results are delivered in several forms. We have published papers and technical reports as listed in Section 4.1, given presentations in conferences, workshops, and specific visits to universities and companies as listed in Section 4.2. We have also released software toolkits on August 1997, as summarized in following subsections.

We also include some references to results produced from the Microlanguages and Immunix projects that have benefitted from or used Synthetix deliverables. The deliverables from these other projects are clearly marked under their own headings.

4.3.2 June–Dec 1994

During the initial six months of the grant we worked on the development of the deliverables, as described below.

4.3.3 Jan–Dec 1995

Specialization Methodology We have been refining the specialization methodology in our work. A preliminary version of the methodology has been described in our SOSP paper [22]. The refined methodology, which includes the use of and support for microlanguages, is under active development.

Specialization Toolkit Our main accomplishment is the creation and refinement of the concept of *specialization class*. Each concrete specialization is an instance of a general type of specialization, defined by the specialization class. Many tools in the specialization toolkit are based on this concept. The explicit (and machine readable) definition of specialization classes allows the software tools to communicate with each other easily. We are writing several papers on specialization classes and developing code using them. Preliminary versions of specific tools have been under test by the members of our group, including the Tempo-C on-line partial evaluator in France and the dynamic kernel module replugger at OGI.

Software Feedback We have been designing and implementing a software feedback toolkit for fine-grain adaptive distributed computation. We have released a distributed multimedia MPEG player in May 1995. There are two recent developments in this effort. First, conceptually, we are modeling software feedback as a fine-grain specialization technique. This way, we will be able to use the specialization tools to support software feedback, and integrate the software feedback toolkit into the specialization toolkit easily. Second, concretely, we are re-implementing the distributed multimedia video/audio player to use the software feedback toolkit. This is a Synthetix deliverable.

Microlanguage Development We have started the development of our first microlanguage for file systems. From the conceptual side, this effort is based on recent work on the specialization of file systems such as our SOSP paper [22] and other research papers on choosing the appropriate caching strategy (e.g., [18]) for an application. ¿From the systems building side, this work is integrated with the construction of the specialization toolkit. We are re-designing the specialization tools to make them extensible and add hooks supporting microlanguages from the beginning.

4.3.4 Jan–Dec 1996

Specialization Methodology We have been refining the specialization methodology in our work. Using the plans for experimentation, we have been designing and implementing the specialization and software feedback toolkit as described below. A new experiment on automated elimination of copying has been reported [30].

Specialization Toolkit We have been refining the concept of specialization class. Each concrete specialization is an instance of a general type of specialization, defined by the specialization class. Currently, the specialization class is considered the "assembly language" of specialization, into which the microprograms will be translated. Components completed for alpha testing include the Replugger [10] the type-based guard checker, and the preliminary version of the Tempo-C compiler [8]. The second paper [12] describes the current refinements of specialization classes. Besides the Replugger [10], the type-based guard checker, and the preliminary version of the Tempo-C compiler [8], we have continued the development of the memory run-time guard checker. All tools are being tested and used in our experimental work.

Software Feedback We have redesigned the software feedback toolkit. Currently, it consists of a component library being written in C++, composition tools, and guarding tools built using the specialization toolkit. The component library will be illustrated using a new demonstration program that has enhanced functionality compared to the distributed multimedia MPEG player we released in May 1995. A Master's thesis was completed [17], on the "Design of a Multimedia Player with Advanced QoS Control".

Microlanguage Development We have continued the development of our first family of microlanguages. The design has been refined using our experience with the implementations of the multimedia player and the file system specialization experiment. The file system microlanguage [24] is under continued development. The family is an integrated set of four microlanguages to describe and program multimedia applications. At the top level, it will be a Quality of Service (QoS) specification microlanguage. The QoS specification is then translated into a media-aware scheduling microlanguage. The data access part of the scheduling microlanguage is then translated into a low level file system microlanguage. The file system microlanguage is finally translated into specialization classes. We are also exploring a variety of scheduling approaches [1, 28] and developing a CPU scheduling microlanguage to be combined with file prefetching, to help achieve multimedia QoS requirements.

4.3.5 Jan–June 1997

Specialization Toolkit We have used the specialization toolkit to specialize production operating system code, both inside the kernel and in system libraries. In [32], we describe the specialization of the Linux operating system signal handler code and `vmalloc`, a well known package that combines several memory allocation policies. In [20], we describe the specialization of SUNOS RPC code. Using the experience gained from these experiments, we have packaged our specialization toolkit, released in August 1997.

Software Feedback We have been developing the software feedback toolkit [6], which is used in a file system [27] and in a mobile multimedia application [15]. The Software Feedback Toolkit is released on the Web as of August 1997. Both the Synthetix Specialization Toolkit and Software Feedback Toolkit are available to the public at URL:

http://www.cse.ogi.edu/DISC/projects/synthetix/toolkit/

Microlanguage Development and Experiments In the application development using the software feedback toolkit [27, 15], we have been developing microlanguages for a systematic interface among system modules. The initial experience is outlined in [24]. The file system prefetching experiment is outlined in [27], where a microlanguage is being designed for controlling the prefetch policy of file blocks for advanced applications such as multimedia and scientific computation. The mobile application in multimedia is outlined in [15, 16], where network device switching happens regularly, with severe strains on the system resource scheduling due to widely differing bandwidths provided by wired (e.g., T1) and wireless (e.g., 9600 baud) connections. The actual experiments using microlanguages will be reported in future papers.

Information Survivability We continued to discuss our approach with colleagues and exchange ideas. This includes Cowan's participation in the SEI workshop on Information Survivability, his presentation at Intel, and Pu's participation in the DCCA panel. We have collected and classified a number of attacks on binary code, e.g., buffer overflow in the stack. We are designing experiments that use variation and diversity, as well as specialization tools such as MemGuard, to handle these attacks.

5 Important Findings

5.1 Early Findings

The Synthetix project consists of a combination of efforts from the operating systems side and from the programming languages side. Our initial important findings have been summarized in two papers, both appearing in the most prestigious conference of each area. In our SOSP paper [22], our most important findings in operating systems are:

1. Systematic specialization is feasible and worthwhile (significant performance gains in the kernel calls studied). Our current work on specialization classes [12] follows this thread.

2. Large-scale specialization requires sophisticated and easy-to-use software tools. See the discussion below.

In our POPL paper [8], our most important findings in the programming languages area are:

1. It's possible to build an on-line partial evaluator (Tempo-C) to support specialization.

2. It's possible to apply the partial evaluator to significant pieces of operating system code written in (cleaned-up) C. Our current experiments [30, 20, 32] follow this thread.

Our experimental work [7, 31, 13, 4] has validated our research ideas and provided directions to its continued development.

5.2 Summary of Project Findings

At a high level, the Synthetix project has confirmed the hypothesis that specialization is a promising technique for improving systems software. At the same time, we have identified challenging research issues in the technology transition of specialization to practice. Specifically, the automation towards the generation of specialized code and guarding of quasi-invariants is clearly the bottleneck in the wide spread adoption of specialization technology. Software tools that help programmers manage the creation and evolution of specialized code are the main contributions of Synthetix.

The Specialization Toolkit consists of the TypeGuard [11], MemGuard, the Replugger [10], and Tempo-C compiler [8] (available on request). The Software Feedback Toolkit consists of a component library for C^{++} and Java, with a re-implementation of the distributed multimedia MPEG player [5] to demonstrate the toolkit components. Both toolkits are available on the Web as of August 1997. These toolkits are being gradually adopted by industry groups such as Intel and Tektronix to facilitate their specialization efforts. We are further developing tools for specific application areas under the Microlanguages, Immunix, Quasar Microfeedback, and Quasar Systemic QoS projects, as well as smaller efforts funded by Intel and Tektronix.

Software in general and operating systems in particular are growing steadily in size, for example by the incorporation of Internet Explorer and Viper transaction processing monitor into the Microsoft Windows NT 5.0. Although dynamic object technologies such as DCOM have postponed a crisis in the integration of these facilities, the performance, reliability, and survivability of these ever growing software systems seem to decline steadily, too. As they mature, specialization tools show increasing relevance towards the management and mitigation of systems software evolution problems.

6 Significant Hardware Developments

The Synthetix project is a software development project. We have not planned any hardware developments. During the project, no unexpected hardware developments were necessary.

7 Special Comments on Industrial Collaboration

In the Synthetix project, we have maintained an active collaboration with industry. Throughout the project, we have the following active collaborations:

- Hewlett-Packard: HP-UX operating system development group funded the initial personnel, software, and hardware experiments. We also benefitted greatly from regular meetings at Cupertino with the developers, who convinced us to develop specialization tools as the means for technology transfer.

- Tektronix: distributed multimedia application support based on software feedback. We are receiving funding from Tektronix for this collaboration. One of the PIs, Jonathan Walpole, is spending his sabatical working with Tektronix.

- Xerox PARC: use of reflection in specialization and the application of specialization as reflection. As our collaborators (G. Kiczales) receive additional funding in the Information Survivability area, we expect closer interaction in this area.

- Intel: application of specialization to distributed multimedia and architectural support for specialization. We received funding in 1996 and 1997 for this collaboration. One of our graduating PhD students, Jon Inouye, has joined Intel and will transfer specialization technology to them.

8 Implication for Further Research

Although the Synthetix grant has ended, the project as a whole is ongoing, with further DARPA funding (The Microlanguages contract, the Immunix project, and the Quasar Microfeedback project) and industrial funding (e.g., Tektronix and Intel). Currently, we are working on the following aspects of the research:

- Specialization Methodology: We are continuing to refine the methodology as we develop the specialization and software feedback toolkits. We are redoing the specialization experiments using the toolkit, as we use the experience to refine the toolkits. Ongoing experiments include specialization of Unix `vmalloc` for performance, and specialization of signal handlers for modularity, and random specialization for generic operating system modules for survivability.

- Specialization Toolkit: We are using toolkit components in our experimental work. The Specialization Toolkit, consisting of the Replugger, the TypeGuard, the MemGuard, and Tempo-C available on request, has been released to the public (Aug. 1997). We are further documenting our toolkit components towards the research on specialization of multi-layer software systems.

- Software Feedback: The new software feedback toolkit continues to be developed, including the component library, composition and guarding tools, and the demonstration distributed MPEG multimedia player program. The composition and guarding tools will use the specialization toolkit wherever feasible. The Software Feedback Toolkit

has been release to the public (Aug. 1997). We are further developing the toolkit as part of the Quasar Microfeedback project (Quorum program).

- Microlanguage Development: We are developing our first family of microlanguages, which consists of a QoS specification microlanguage, a resource scheduling microlanguage, a file system microlanguage, and appropriate specialization classes in the kernel. Several examples of microlanguage has been designed for file systems and mobile applications. This is in the context of the Microlanguages project.

- Information Survivability: We are designing the experimental evaluation of specialization as a defense against system attack methods we have collected. For example, many of the attacks use the overflow of buffer allocated on the program stack. We are using diversity and existing specialization tools such as MemGuard for experimentation. Another area of system protection introduced by Immunix is in the Morphing File System, which adds dynamic checking into the file system sharing. This is in the context of the Immunix project (Information Survivability program).

References

[1] V. Baiceanu, C. Cowan, D. McNamee, C. Pu, and J. Walpole. Multimedia applications require adaptive CPU scheduling. In *Proceedings of the 1996 Workshop on Resource Allocation Problems in Multimedia Systems*, Washington, D.C., December 1996.

[2] B.N. Bershad, C. Chambers, S. Eggers, C. Maeda, D. McNamee, P. Pardyak, S. Savage, and E.G. Sirer. SPIN - An Extensible Microkernel for Application-specific Operating System Services. In *SIGOPS 1994 European Workshop*, February 1994. UW Technical Report 94-03-03.

[3] A. Black and J. Inouye. System support for mobility. In *Proceedings of the Seventh ACM SIGOPS European Workshop*, pages 129–132. ACM, September 1996.

[4] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. Demonstrating the effect of software feedback on a distributed real-time MPEG video audio player. In *Proceedings of the 1995 ACM International Conference on Multimedia*, San Francisco, November 1995.

[5] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A distributed real-time MPEG video audio player. In Tom Little, editor, *Proceedings of the 1995 International Workshop on Network and Operating System Support for Digital Audio and Video*, volume 1018 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995. Also appeared in the Proceedings of the 1995 International Workshop on Network and Operating System Support for Digital Audio and Video, April 1995, New Hampshire.

[6] Shanwei Cen. *A Software Feedback Toolkit and its Applications in Adaptive Multimeida Systems*. PhD thesis, Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology, August 1997.

[7] C. Consel, L. Hornof, F. Noel, and E.N. Volanschi. A uniform approach to compile-time and run-time specialization. In *Proceedings of the 1996 Dagstuhl Workshop on Partial Evaluation*, Germany, February 1996.

[8] C. Consel and F. Noel. A general approach for run-time specialization and its application to c. In *Proceedings of the 1996 ACM Symposium on Principles of Programming Languages*, Florida, January 1996.

[9] C. Consel, C. Pu, and J. Walpole. Incremental partial evaluation: The key to high performance, modularity and portability in operating systems. In *Proceedings of the 1993 ACM Symposium on Partial Evaluation and Semantics-Based Program Manipulation*, Copenhagen, Denmark, June 1993.

[10] C. Cowan, T. Autrey, C. Krasic, C. Pu, and J. Walpole. Fast concurrent dynamic linking for an adaptive operating system. In *Proceedings of the International Conference on Configurable Distributed Systems*, Maryland, May 1996.

[11] C. Cowan, A. Black, C. Krasic, C. Pu, and J. Walpole. Automated guarding tools for adaptive operating systems. Technical Report OGI-CSE-95-0XX, Department of Computer Science and Engineering, Oregon Graduate Institute, May 1996.

[12] C. Cowan, A. Black, C. Krasic, C. Pu, J. Walpole, C. Consel, and N. Volanschi. Specialization classes: An object framework for specialization. In *Proceedings of the 1996 International Workshop on Object-Orientation in Operating Systems*, Seattle, October 1996.

[13] C. Cowan, S. Cen, J. Walpole, and C. Pu. Adaptive methods for distributed video presentation. *ACM Computing Surveys*, 27, December 1995.

[14] C. Cowan and H. Lutfiyya. A wait-free algorithm for optimistic programming: Hope realized. In *Proceedings of the 16th International Conference on Distributed Computing Systems*, Hong Kong, May 1996.

[15] J. Inouye, Cen S., C. Pu, and J. Walpole. System support for mobile multimedia applications. In *Proceedings of the Seventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*, St. Louis, May 1997.

[16] J. Inouye, J. Walpole, and J. Binkley. Physical media independence: System support for dynamically available network interfaces. Technical Report OGI-CSE-97-001, Department of Computer Science and Engineering, Oregon Graduate Institute, January 1997.

[17] Rainer Koster. Design of a multimedia player with advanced QoS control. Master's thesis, Oregon Graduate Institute of Science and Technology, Portland, Oregon, 1996.

[18] D. Kotz. Disk-oriented I/O for MIMD multiprocessors. In *Proceedings of the First Symposium on Operating Systems Design and Implementation*, pages 61–74, Monterrey, CA, November 1994. Usenix.

[19] H. Massalin and C. Pu. Threads and input/output in the Synthesis kernel. In *Proceedings of the Twelfth ACM Symposium on Operating System Principles*, pages 191–201, Arizona, December 1989.

[20] G. Muller, R. Marlet, E-N. Volanschi, C. Consel, C. Pu, and A. Goel. Fast, optimized sun rpc using automatic program specialization. Technical Report IRISA PI-1094, IRISA/University of Rennes, January 1997.

[21] S. O'Malley and L. Peterson. A dynamic network architecture. *ACM Transactions on Computer Systems*, 10(2):110–143, May 1992.

[22] C. Pu, T. Autrey, A. Black, C. Consel, C. Cowan, J. Inouye, L. Kethana, J. Walpole, and K. Zhang. Optimistic incremental specialization: Streamlining a commercial operating system. In *Proceedings of the Fifteenth Symposium on Operating Systems Principles*, Colorado, December 1995.

[23] C. Pu, A. Black, C. Cowan, and J. Walpole. A specialization toolkit to increase the diversity of operating systems. In *Proceedings of the 1996 ICMAS Workshop on Immunity-Based Systems*, Nara, Japan, December 1996.

[24] C. Pu, A. Black, C. Cowan, and J. Walpole. Microlanguages for operating systems specialization. In *Proceedings of the 1997 POPL Workshop on Domain-Specific Languages*, Paris, January 1997.

[25] C. Pu, H. Massalin, and J. Ioannidis. The Synthesis kernel. *Computing Systems*, 1(1):11–32, Winter 1988.

[26] C. Pu and J. Walpole. A case for adaptive OS kernels. In *Proceedings of the 1994 OOPSLA Workshop on Flexibility in Systems Software*, Portland, Oregon, October 1994.

[27] D. Revel, C. Cowan, D. McNamee, C. Pu, and J. Walpole. Predictable file access latency for multimedia. Department of Computer Science and Engineering, Oregon Graduate Institute; March 1997.

[28] D. Revel, C. Cowan, D. McNamee, C. Pu, and J. Walpole. An architecture for flexible multimedia prefetching. In *Proceedings of the 1996 Workshop on Resource Allocation Problems in Multimedia Systems*, Washington, D.C., December 1996.

[29] H. Thimm, W. Klas, J. Walpole, and C. Pu, C.and Cowan. Managing adaptive presentation executions in distributed multimedia database systems. In *Proceedings of the IEEE International Workshop on Multimedia Database Management Systems*, pages 152–159, Blue Mountain Lake, NY, August 1996.

[30] E.-N. Volanschi, G. Muller, C. Consel, L. Hornof, J. Noyé, and C. Pu. A uniform automatic approach to copy elimination in system extensions via program specialization. Research Report 1021, Irisa, Rennes, France, June 1996.

[31] E.N. Volanschi, G. Muller, and C. Consel. Safe operating system specialization: the rpc case study. In *Proceedings of the First Workshop on Compiler Support for Systems Software*, Arizona, February 1996.

[32] J Walpole, Cen S., C. Cowan, R. Koster, D. Maier, C. Pu, and L. Yu. Adaptive real-time video streaming over the internet. In *Proceedings of the 26th Applied Imagery Pattern Recognition Workshop (AIPR-97)*, Washington, DC, October 1997.