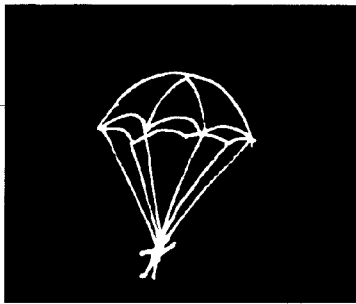




Experience with Formal Methods in Critical Systems

SUSAN GERHART, *University of Houston at Clear Lake*

DAN CRAIGEN, *ORA Canada*

TED RALSTON, *Ralston Research Associates*

◆ *Although there are indisputable benefits to society from the introduction of computers into everyday life, some applications are inherently risky. Worldwide, regulatory agencies are examining how to assure safety and security. This study reveals applicability and limitations of formal methods.*

Several regulatory and standards organizations in North America and Europe are investigating how to certify critical systems, and formal methods are figuring importantly in their studies. Several draft standards and regulations are circulating, covering many application domains and approaches — from the mandatory use of formal methods to provisions that recommend “best practices.”

To better understand the contributions of formal methods, we systematically studied 12 cases in which organizations in North America and Europe used formal methods. This six-month study, sponsored by the US National Institute of Science and Technology, the US Naval Research Laboratory, and the Atomic Energy Control Board of Canada, is detailed in the box on pp. 22-23.

We had three main objectives:

◆ To better inform deliberations with-

in industry and government, especially within the study's sponsors, on standards and regulations.

◆ To provide an authoritative record on the practical experience of formal methods to date.

◆ To suggest where more research and technology development are needed.

This article focuses on our findings when formal methods were applied to safety- and security-critical systems within a regulatory environment. We also impart some lessons learned about applying formal methods in other environments.

In the context of this article, formal methods are mathematical synthesis and analysis techniques used to develop computer-controlled systems. The overriding reason to develop and use formal methods is to attempt to predict, in a scientific manner, the behavior of computer-controlled systems.

As our study shows, formal methods

span requirements capture, reengineering, documentation, testing, and other forms of analysis. We also found that, although formal methods can help develop more predictable systems, developers, regulators, and the public should understand that formal methods — like any technology — have limits when applied to critical applications.¹ There are both practical and theoretical limits and much room for more research.

We also found significant technology-transfer problems. In some cases, the use of formal methods was a one-shot effort, with little follow-on, and nowhere did we find any widespread penetration.

CASE STUDIES

In keeping with our objectives, we provided both experiential data and an analysis of that data. The 12 projects we studied were drawn from commercial, exploratory, and regulatory domains.

◆ *Commercial cases* involve products that must be commercially viable. We studied five commercial projects, briefly described in the box on p. 24.

◆ *Exploratory cases* are investigations of the potential use of formal methods in a particular setting. We studied three ex-

ploratory projects, also briefly described in the box on p. 24.

◆ *Regulatory cases* exhibit safety- or security-critical attributes and thereby attract the attention of standards communities and agencies and regulators. These cases are elaborated in the pages following this article:

1. *Darlington Nuclear Generating Station.* At this power plant outside Toronto, developers used formal methods to help achieve certification for the plant's shutdown systems, the first software-driven shutdown systems in Canada.

2. *Paris Metro Signaling System.* This system, designed to reduce the separation between trains in the Paris Metro, is the first safety-critical software system certified by the French railway authority. Its developers used formal methods extensively for verification and validation.

3. *Traffic Alert and Collision Avoidance System.* To improve an industry-government working group's capability to review this system, which alerts and advises pilots in the event of an impending collision, formal specifications of two major subsystems were developed using a formal, graphical notation.

4. *Multinet Gateway System.* The use of formal methods was required in develop-

ing this security-critical internetwork gateway, to achieve clearance from the US Department of Defense to carry sensitive information.

In addition to the two-volume official report,²⁻³ we have written of this study elsewhere, to address issues in software engineering⁴ and formal methods research and development.⁵ Volume 1 of the final report describes the study, formal methods, the cases studied, our approach, and our analysis, findings, and conclusions. Volume 2 provides case-study details.

For each case, we present a description, summarize the information obtained (from interviews and the literature), provide an evaluation of the case, highlight R&D issues pertaining to formal methods, and provide some conclusions.

Together, the volumes total 300 pages. It is impossible to condense such a prodigious amount of technical information into an article of this length and preserve all the subtleties. We have tried to capture the main themes here, but urge that you obtain a full report if you have any questions.

LESSONS LEARNED IN REGULATORY CASES

Regulators can certify the product (as

OUR APPROACH TO STUDYING FORMAL-METHODS EXPERIENCE

Previous studies, some of which we have worked on, have uncovered anecdotal evidence of a significant increase in the use of formal methods.¹⁻² We decided to team up and further investigate this evidence. Under Ted Ralston's leadership, we found sponsors and began our survey, which took about 1.5 person-years.

Informative acquisition. Our general procedure was to

1. Send an initial questionnaire to the interview subjects.
2. Study relevant project literature.
3. Interview project participants using a second question-

naire to structure the interviews. We conducted 23 interviews, which involved about 50 individuals in North America and Europe (we interviewed at least two and as many as five project participants). Each interview took anywhere from 30 minutes to one and a half days.

4. Produce raw notes on the interview. These notes, along with information gathered from the literature, supplemented the initial and interview questionnaires. The study team wrote a report on each case, using a reasonably consistent framework. We sent the individual case reports to project participants for

comment and corrections.

5. Analyze the cases. As we went along, we developed an analytic framework.

Questionnaires. Both questionnaires had six categories:

◆ *Organizational context.* We were intentionally vague as to the meaning of "organization" because we wanted respondents to define the environment they felt had the greatest impact on them.

◆ *Project content and history.* We asked for a description of the application, its evolution, and the resources used. Other questions in this category also helped us compare organizational and pro-

ject compositions.

◆ *Application goals.* We asked why the application was developed and the major influences that led to or affected development.

◆ *Formal-methods factors.* These questions elicited the "why" and "what" of the project's formal-method selection process.

◆ *Formal methods and tool use.* We asked what processes and tools were used to support the chosen formal method.

◆ *Results.* We asked for general conclusions about the project and the effect of formal methods.

Analytic framework. For each case, we produced vectors per-

the US Federal Aviation Administration has done with its system to avoid mid-air collisions), the *process* (as is the Atomic Energy Control Board of Canada's usual practice), and the *professional* (as the UK Ministry of Defence's Interim Standard on the Procurement of Safety Critical Systems⁶ does).

The organizations charged with developing and applying these standards and regulations are concerned with three issues:

- ◆ What is the best policy approach to achieve assurance?
- ◆ What should the requirements be and how can they best be achieved reasonably?
- ◆ How can conformance with certification requirements best be demonstrated?

In the regulatory cases, organizations seemed to use formal methods primarily to unambiguously and comprehensibly demonstrate to regulators a system's proposed effects (do they satisfy requirements?) and to produce evidence (not only through test plans, but by proof) that the code conforms to its requirements. In addition, if formal mathematical models are used, developers must definitively demonstrate the relationship between the model and the underlying code.

As the box below explains, we developed a list of features against which to evaluate the effect of formal methods. For each feature, we asked, "Was the influence of formal methods positive, neutral, or negative (or was there enough information to discern)?" Our full report analyzes the factors that led to our conclusions.

Client satisfaction. *Were clients happier with the product?* We recognize, of course, that happiness is influenced by many factors, including enhanced reliability and reduced cost.

In all four cases, we found that the clients (chiefly the regulatory agencies) were satisfied with the products developed using formal methods. Pivotal to client satisfaction were perceived gains in assurance and comprehension.

We found that mathematical-based modeling made system behavior more comprehensible. Everyone we inter-

viewed on the Multinet project said they better understood its security aspects because formal methods were used. The FAXs working group said the formal description of the Collision-Avoidance System's Logic subsystem was easier to review and modify than the original pseudocode and natural-language specifications.

Product cost. *Was the overall cost of the product reduced or profit increased?*

We were unable to obtain sufficient information to draw strong conclusions about the cost of using formal methods on these four projects. Regulatory systems are inherently expensive because the products are so impor-

tant and involve so many organizations. We can say that the cost of using formal methods was, with the exception of Multinet, where they were required, just small change relative to total cost.

We did not find that product costs dra-

PERCEIVED GAINS IN COMPREHENSION AND ASSURANCE WERE PIVOTAL FACTORS IN WHETHER CLIENTS WERE SATISFIED WITH PRODUCTS DEVELOPED WITH FORMAL METHODS.

taining to important product features and process activities. Relative to the organization's usual approach to development (and this may be a subjective measurement), we characterized whether the use of formal methods played a positive, neutral, or negative role. When no information on a product feature or process characteristic was obtainable, we did not make an entry.

The features and activities included as part of our vectors, described in the main text, are not necessarily independent of each other. For example, client satisfaction depends on product cost and quality. What we are trying

to measure is the relative effect of using formal methods.

We also included an analysis of how the project started, the types of events that either helped or hindered it, and its status.

Biases and limits. As experts in formal methods, each of us, to differing extents, have vested interests in the technology. One way we addressed this was to form a committee who reviewed and commented on the interim reports. This also gave us input from individuals who have expertise that complements ours.

Our methodology was a common-sense approach—as op-

posed to a specific social-scientific approach—driven by the need to discover both the general context of how formal methods are used and specific aspects of how a method was selected and how it was used. Many variables affected the success or failure of the cases, so our ability to reach purely scientific conclusions is limited.

Other limits were

- ◆ We did not have much time or money.
- ◆ We did not study some important classes of methods (for example, standard protocol languages like Specification Design Language and Lotus) and some important domains.

◆ Although every project we studied was successful to various degrees and in different ways, we did not seek examples of outright failure. Such cases—there are probably dozens of them—would probably be manifested in drop-outs and personnel turnover.

REFERENCES

1. *Formal Methods for Trustworthy Computer Systems*, D. Craigen and K. Summerskill, eds., Springer-Verlag, London, 1990.
2. S. Gerhart et al., *Formal Methods Transition Study Final Report and Videotape*, Tech. Report TR STP-FT-322/323-91, MCC Software Technology Program, Austin, Tex., 1991; available from RICE, University of Houston at Clear Lake.

COMMERCIAL AND EXPLORATORY CASES

In addition to the four regulatory projects that are the focus of this article, we studied five commercial projects and three exploratory projects.

SSADM tool set. The British firm Praxis developed a computer-aided systems-engineering tool set to support the use of the Structured Systems Analysis and Design Method.¹ Praxis used Z to formally specify the tool set's infrastructure. Later, the project team developed guidelines for the use of Z with object-oriented design. The project resulted in 37,000 lines of Objective C and a 350-page, annotated Z specification.

Customer Information Control System. CICS is a large transaction-processing system developed by IBM. Developers at IBM Hursley, UK, reengineered a major portion of CICS Version 3 Release 1 and subsequent releases using the Z method and tools. CICS is approximately 800,000 lines of source code; of the approximately 50,000 lines of new or modified code, 37,000 were completely specified using Z and about 11,000 were partially specified using Z. IBM claims that using Z reduced both development cost and error rates.²

Cleanroom applications. To better understand the Cleanroom methodology, we investigated two industrial applications: one at the US National Aeronautics and Space Administration's Goddard Space Flight Center and another at IBM Federal Systems Division. The Goddard application was a system to provide attitude ground support for NASA's International Solar Terrestrial Physics Satellite. The IBM application, the focus of our Cleanroom study, was the development of a Cobol Structuring Facility to convert old Cobol

programs to a semantically equivalent "structured-programming" form.³ Cobol/SF comprises 80,000 lines of code and required 70 person-months. It demonstrated to IBM management the potential of the Cleanroom methodology.

Software Architecture for Oscilloscopes using Z. Tektronix, in Beaverton, Oregon, used Z to develop a reusable software architecture for several new oscilloscope products.⁴ Developers used Z to explore design ideas through mathematical models, which were viewed as nonexecutable prototypes. Their intent was not formal correctness; they used Z as a design aid. The software architecture consists of 200,000 lines of code and 30 pages of Z.

Inmos transputers. Inmos, in Bristol, England, makes a transputer family of 32-bit VLSI circuits that have a unique architecture designed for concurrent, multiprocessor applications (the processor, memory, and communication channels are self-contained on each transputer chip). In 1985, a small group of Inmos designers began exploring how to use formal program specification, transformation, and proof techniques to design microprocessors. We studied three related projects, all of which use formal methods in some aspect of the design or development of components of three generations of the Inmos transputer: the use of Z to specify the IEEE floating-point standard which was applied to two successive generations of transputer (a software and a hardware implementation);⁵ the use of Z and Occam to design a scheduler for the T800 transputer; and the use of Communicating Sequential Processes and Calculus of Communicating Systems plus a "refinement checker" in the design and verification

of a new feature of the T9000 transputer, the Virtual Channel Processor.

Large correct systems. The Lacos project seeks to transfer the Raise (Rigorous Approach to Industrial Software Engineering) language and tool set into practice. Raise, which evolved from the Vienna Development Method, is being commercialized by Computer Resources International of Denmark — the producer partner. We interviewed one consumer partner at length and a second briefly. The first, Lloyd's Register, is evaluating Raise (as well as other methods) for a data-acquisition and equipment-management system. The second, Matra Transport, builds railway and other systems. Both are building consultancies in formal methods and assessment. Because Lacos is a large, on-going, technology-transfer project, it is an important one to follow for the remaining three years of ESPRIT funding and beyond.

Token-Based Access Control System. TBACS is a smartcard access-control system with cryptographic authentication being developed at the US National Institute of Standards and Technology. One group at NIST had been developing a series of prototype smartcards for cryptographic authentication for network access. A second group was looking at new technology to support open systems and other commercial standards. A member of the second group, looking for an application on which to base an experiment in formal methods, chose the smartcard application. In this case, the researchers chose a tool set and approach that followed the standard process in trusted-system certification, using a theorem-prover to verify that a design

meets the requirements of a security-policy model.⁶

Hewlett-Packard medical instruments. Although this was a large, important project, we consider it to be exploratory because its primary objective was technology transfer. The product is a real-time database, the Analytical Information Base, of patient-monitoring information. Using an HP-developed specification language (a VDM variant), a formal-methods transfer group and a developer produced a specification. The transfer effort failed because time-to-market was the key feature, and formal methods offered little beyond the high quality already achievable by other means that were consistent with the culture of the organization.

REFERENCES

1. D. Brownbridge, "Using Z to Develop a CASE Toolset," *Proc. Z Users Workshop*, Springer-Verlag, London, 1989, pp. 142-149.
2. I. Houston and S. King, "CICS Project Report: Experiences and Results from the use of Z," *Proc. VDM 91*, Volume 551, Springer-Verlag, Berlin, 1991, pp. 588-596.
3. R. Linger and H. Mills, "A Case Study in Cleanroom Software Engineering: the IBM COBOL Structuring Facility," *Proc. Campsac*, IEEE CS Press, Los Alamitos, Calif., 1988, pp. 10-17.
4. D. Garlan and N. Delisle, "Formal Specifications as Reusable Frameworks," *Proc. VDM 92*, Springer-Verlag, Berlin, 1990, pp. 150-163.
5. G. Barrett, "Formal Methods Applied to a Floating Point Number System," *IEEE Trans. Software Eng.*, 1989, pp. 611-621.
6. D.R. Kuhn and J.F. Dray, "Formal Specification and Verification of Control Software for Cryptographic Equipment," *Proc. Computer Security Applications Conf.*, IEEE CS Press, Los Alamitos, Calif., 1990, pp. 32-43.
7. *Hewlett-Packard Journal* special issue on HP-SL, Dec. 1991, pp. 24-65.

matically increased, but the effects of formal methods were measured in other ways — namely, something was needed to achieve assurance. In Darlington, perhaps the costs of use were higher than they should have been because there were no tools and the methodology was immature. The cost of using formal methods in the TCAS project will be measured over time as it is upgraded.

Product impact. *Was the product important to the company's profit margin or an organization's reputation?*

In general, we found that the surveyed products developed using formal methods positively affected the organizations involved. Perhaps the clearest case is the Paris Metro: Not only did the project achieve its goal of eliminating the need for an additional railway line, but project leader GEC Alsthom is using its new-found expertise to market its services.

For TCAS, independent verification and validation uncovered a handful of consequential problems with the CAS Logic pseudocode, which are being repaired for the next release.

Product quality. *Was the quality of the product improved by the use of formal methods? By "quality," we mean good safety and security properties, enhanced functionality and performance, and fewer errors.*

We found evidence of increased quality, but in different forms. To the Multinet and Paris Metro developers, formal methods helped achieve quality goals. On the Multinet project, mathematical analyses and modeling improved the understanding of the security principles. The Paris Metro system met the Paris regional rail system's quality requirements, even though it had to meet increased functionality requirements.

Darlington is an interesting case because formal methods did improve assurance, but the regulators did not claim an increase in quality. In effect, formal methods were used a posteriori to model what existed and to demonstrate that the code met the requirements.

The CAS Logic formalism illustrated subsystem requirements more clearly than the pseudocode, and independent

verification and validation uncovered consequential errors in the pseudocode.

Time to market. *Was the product, or family of products, made available for marketing more rapidly (or at least not delayed further)?*

Time to market does not seem to have been a primary concern in these projects, but the possible effects of failing to complete product development or to achieve licensing can be extensive. This was especially obvious in the Darlington project; the operator of the plant would have had to revert to a hardware solution at a cost of about \$1 million and a one-year wait for parts had they not been able to certify the software-based shutdown systems.

On the Paris Metro project, failure to achieve the time-to-market goal would have cost hundreds of millions of dollars for an additional railway line.

Process cost. *This feature measures cost in terms of reduction in effort and is based on the adage "time is money."*

We found the effort to attain assurance and comprehension in regulatory cases to be substantially more than we found in the commercial cases.

Darlington reported that formal methods cost about \$4 million; the Paris Metro project invested about 120,000 person-hours. These costs are substantially more than we would expect in future projects. In both cases, developers had to develop not only the product but also the technology. GEC Alsthom has indeed reported that the cost of applying formal methods has been lower on the two projects they undertook after the Paris Metro.

Process impact. *What effect did the formal-methods process have on the organization?*

In two cases, we rated the effect as positive: Both the Paris railway system and the FAA participants have expressed an in-

terest in the continued use of formal methods.

In the other two, the effect was neutral: The developers of the Darlington shutdown systems have acknowledged that formal methods must play an important role in future developments, but some managers now have less enthusiasm for shifting from hardware to software controllers. And the effect of formal methods on the process used by the Multinet developers is lessened by the contract-driven nature of the organization — novelty is not always viewed positively in responding to proposal requests.

Pedagogical. *What did the organization make of this learning opportunity?*

All the organizations reported receiving educational benefits from the formal-methods efforts. They better understand formal methods and research-and-development issues. And, especially in the Darlington and Multinet cases, they better understand what process will satisfy regulators. The Darlington and TCAS projects have produced substantial feedback to research programs.

Tools. *Did formal-methods tools help or hinder the development of the product? Were they reliable?*

We found that there does appear to be a need for automated deduction support, especially to demonstrate consistency of code and specification. There is an apparent dichotomy in that the Multinet and Paris Metro projects used formal-methods tools extensively; whereas, Darlington and TCAS didn't. This dichotomy is more in appearance than in reality, because the developers in the Darlington and TCAS projects voiced no philosophical objections to tools, there simply weren't any available. Tool development is the subject of ongoing research.

Design. *Were the designs produced using*

WE FOUND THERE DOES APPEAR TO BE A NEED FOR AUTOMATED DEDUCTION SUPPORT, ESPECIALLY TO DEMONSTRATE CONSISTENCY.



formal methods fundamentally better or worse in some respect? For example, were they simpler?

Formal methods appear to have uniformly improved either the product or the developers' understanding of it. Consequently, they had more faith in its quality than with traditional development methodologies. These improvements appear to be the result of using mathematics to analyze domain theories (Multinet), mathematical arguments to demonstrate code and specification properties (Paris Metro), and mathematical-based notations to describe requirements (TCAS) and specifications (Darlington).

Reusable components. *Did the use of formal methods ease the development or use of reusable hardware, software, designs, or abstractions?*

In the two cases for which we could draw a conclusion, formal methods positively affected reuse. The security and safety models developed as part of the Multinet and Paris Metro projects, respectively, are reusable. The reuse benefits involved models or mathematical theories, not code. However, although we found no evidence of code reuse, we believe formal methods will ultimately help develop reusable code because the resulting specifications will unambiguously describe the code's functionality and because using mathematics will result in sufficiently general code that is more suitable for reuse.

Maintainability. *Is it easier to maintain the product (including incremental updates and error repair)?*

It appears that formal notations to capture specifications and requirements, information hiding, and modularization should increase comprehension and clarify interactions between parts of systems, thereby making maintenance simpler. However, if the benefits are to be ongoing,

maintenance and product improvements must use and build on practices already in place.

Requirements capture. *Was it easier to acquire requirements?*

We were told by the people involved in all four cases that formal methods improved intellectual control, clarified requirements, and removed superfluous requirements. For example, the FAA's working group believes that the University of California at Irvine's Statecharts-like notation has allowed them to argue about the technical substance of the formal descriptions, instead of minor grammatical matters. The Multinet developers reported a sharper understanding of the security principles involved in building a heterogeneous network.

Verification and validation. *What effect did formal methods have on overall verification and validation?*

Using formal methods was crucial to achieving assurance in these systems. Proofs played a fundamental role in the Darlington and Paris Metro projects. Some projects also used formal descriptions to generate test cases and to complement other verification-and-validation procedures. We believe strongly that developers must use a spectrum of such techniques to attain adequate assurance.

GENERAL LESSONS LEARNED

Because the support and market for formal methods will eventually transcend organizational and application domains, we believe it is instructive to summarize the lessons learned from all 12 case studies.

Wider applicability. Ten years ago, most believed that investing in the technology and know-how to apply formal methods was warranted in only a few domains, like

computer security. We found that two major changes have occurred since then:

- ◆ Formal methods now encompass not only program verification but also formal specification, formal design, and similar terms that connote the use of a mathematical description and analysis.

- ◆ Formal methods are now being used in a range of applications by organizations who require more intellectual control than traditional development methods can offer. We found formal methods being used to develop oscilloscopes, ships, satellites, smart cards, transaction-processing units, floating-point arithmetic units, networks, medical instruments, programming environments, and language processors.

Larger scale. Formal methods are being applied to systems of significant scale and extreme importance, although project success depends on more than formal methods. We learned that lines of code, the most common measure, does not always correlate with scale. Not only is lines of code an awkward measure (because of differences in formats and the inclusion of comments), but it

- ◆ *Does not take into account the use of abstraction.* Abstraction lets us express a family of products in a single formal specification, apart from design and code. It is especially important for effective communication among stakeholders (for example, designers to managers, or among designers from different disciplines).

- ◆ *Does not adequately reflect the design and verification process.* This process goes from safety studies, through many design iterations, often across several organizational boundaries, culminating in what may be a few thousand lines of critical code. It invariably takes a long time and involves many people.

- ◆ *Does not practically represent complexity.* Complexity may have more to do with the environment (rare events, multiple failures, or feature interference) than the code, which could be straight-line but have lots of formulas.

Primary manifestations. We found that formal methods are manifested in many ways

- ◆ Specification only, to capture ab-

WE WERE TOLD IN ALL FOUR CASES THAT FORMAL METHODS IMPROVED INTELLECTUAL CONTROL AND CLARIFIED REQUIREMENTS.

stractions of product families or requirements models.

- ◆ Specification with refinement through multilevel designs.

- ◆ Proofs, sometimes of specification properties, sometimes of refinement steps, and in some cases at or close to code level.

- ◆ Reviews of specifications and design refinements, both the steps performed and other qualities (complexity and clarity) shown by the formal review process.

As we expected, we found that no single formal method is sufficiently general to cover all the significant characteristics of an applications' domain. However, there is generally a weak understanding of how to use different formal methods together or with other methods, like structured analysis.

Use in certification. The first step toward the use of formal methods within a regulatory process is to recognize their potential. The next step is to identify candidate methods — we found that this is very opportunistic. Then the chosen method must evolve to meet the needs of the various stakeholders. The four regulatory cases showed considerable interaction among regulators, developers, and researchers, who are working together to develop practical, effective techniques for system certification.

The long-term effect of this experience will be evident in the standards that emerge and, although no standard has yet been sanctioned on the basis of such cases, de facto standards may arise from high-profile cases like these.

Tool support. To be fully industrialized, formal methods will require computer-based tools, but we found such tools are neither necessary nor sufficient for success. Instead, the dominant drivers are flexibility, mixed modes of communication, and intellectual control. On the other hand, projects that have employed advanced tools (like theorem provers) have not shown inordinately high costs.

One reason current formal methods don't have adequate support technology is that there is no common architecture for the development of scalable tools. There are very large tools and very small ones,

and not much in between. Few products enjoy any commercial support in terms of documentation, training, and evolution, but we expect some of the internal tools used in the project we studied may be commercialized in the next few years.

Technology transfer. Several North American organizations and many more European ones have formal-methods technology-transfer efforts in progress. However, even organizations with successful applications and significant technology-transfer efforts show only a small degree of penetration.

We found three specific technology-transfer efforts:

- ◆ The Lacos (Large, Correct Systems) project, funded for the next several years by the European Strategic Program for Research in Information Technology, is specifically designed to encourage technology transfer. It focuses on a specific tool, Raise.

- ◆ The Cleanroom method is supported by the IBM Software Technology Center, which is chartered to perform training and demonstrations for other parts of IBM.

- ◆ Hewlett-Packard used its Applied Methods Group as its technology-transfer mechanism, but did not achieve critical mass and subsequent support within its developer shop.

In addition, some security applications have been successful, but their results were sensitive and hence not widely reported. Many tools developed under the auspices of security organizations have been prohibited from transfer.

We observed four important factors in transferring formal methods:

- ◆ Sometimes, projects enjoyed greater success when they packaged formal methods as part of a larger methodology. In some cases, the other aspects were more successful than the formal methods, which served as a "carrier" of modern concepts.

- ◆ Successful projects usually had a guru or some form of sustained guidance.

- ◆ Management support at the right time is key.

- ◆ Sometimes the impetus for the adoption of formal methods is a major change in technology (from hardware- to software-based control, for example).

THE MATH INVOLVED IN MOST FORMAL METHODS IS ELEMENTARY. THE GREATER CHALLENGE MAY LIE IN TEACHING USERS HOW TO MODEL SYSTEMS PROPERLY.

Skill building. Skills are building slowly in organizations that are trying to use formal methods on industrial projects. Of the companies we surveyed, the largest number of persons skilled in using formal methods was at IBM: Between 40 to 50 people at Hursley and about the same at Federal Systems Division.

Most of the projects we studied had fewer than 10 knowledgeable project members, but overall the project members had

strong mathematical and engineering backgrounds. Many of those new to the mathematics underlying formal methods (such as predicate calculus) could adapt to formal notations when they were presented suitably. The mathematics involved in most formal methods is elementary, so the greater challenge may lie in teaching users how to model systems properly and carry a design through.

Code-level application. We found that there are limits to applying formal methods at the code level. Most programming languages (and many specification languages) lack adequate semantic bases to fully support formal methods. Developers can use some of the static-analysis and compiler-type tools to complement design refinements down to a level close to code level. However, many aspects of runtime environments and hardware are not treated adequately and must be viewed as holes in the general use of formal methods. Also, refinement is not viewed as cost-effective for commercial applica-

tions, although it may be useful in regulatory cases.

Cost modeling. It is difficult to establish the value of formal methods in situations when there are other methods that can increase quality or when time-to-market dominates product goals. We did not have adequate metrics to perform the cost and cost-benefit analyses that would capture the effect of formal methods on the development process.

Some of the cases we studied attempted to collect performance data using accepted metrics, but we found these to be inadequate. First, the metrics rely heavily on comparisons with previous data collected using the same metrics, which, with one exception, had not been kept. Second, formal methods, as practiced in most cases, require substantial change to the development process as compared to other, informal methods. The metrification methods did not seem to account for these changes.

We believe the 12 cases we studied demonstrate that formal methods are applicable to industrial software development and should not be ignored by industry. Our study found that formal methods are used primarily for

- ◆ **Assurance.** Some projects that require a high degree of confidence, with auditable information, have explicit targets of low or zero error rates.

- ◆ **Domain analysis.** Some projects use formal methods to better understand an application domain.

- ◆ **Communication.** When communication among system stakeholders is the primary need, the use of formal methods is not so much for mathematical analysis, but to supplement informal notations with formal notations.

- ◆ **Evidence of best practice.** Sometimes the use of formal methods bestows a competitive advantage or is simply required.

- ◆ **Reengineering.** Formal methods can be used on products undergoing long-term upgrades and on those that require structure recovery or function enhancement.

For a limited time, our final report is available electronically, on onnemo.ncsl.nist.gov, for instructions, write to survey@ora.on.ca.

The transfer of formal-methods tech-

nology from academia to industry (and vice versa) is slowly proceeding. The industrial application of the technology is constrained by the sophisticated nature of the technology, negative connotations, and the small cadre of experts.

We would also like to see this kind of

systematic study, which focuses on both technical and organizational facets, applied to other technological areas. Such studies — we might call them research-usage feedback studies — provide useful, pragmatic feedback and could serve as another model of research activity. ♦

ACKNOWLEDGMENTS

This study was sponsored by the US National Institute for Standards and Technology, the US Naval Research Laboratory, and the Atomic Energy Control Board of Canada. The US National Science Foundation also provided independent research time for Gerhart for work initiated by Applied Formal Methods, Inc. Previous studies of trends in commercial and regulatory applications of formal methods were performed in conjunction with the organization of the FM89 conference and the Microelectronics and Computer Technology Corp.'s Formal Methods Transition Study. Gerhart's work on this article was done under US National Aeronautics and Space Administration cooperative agreement 9-1-6.

We thank the interviewees for their time, professional information, and openness. Robin Bloomfield participated in many of the Darlington interviews. The advisory committee — Adele Goldberg, John Marciniak, Morven Gentleman, Lorraine Duvall, and John Gannon — and the sponsors' review teams were especially helpful.

REFERENCES

1. *Formal Methods for Trustworthy Computer Systems*, D. Craigen and K. Summerskill, eds., Springer-Verlag, London, 1990.
2. D. Craigen, S. Gerhart, and T. Ralston, "An International Survey of Industrial Applications of Formal Methods, Volume 1 Study Methodology," Tech. Report PB93-178556/AS, National Technical Information Service, Springfield, Va.; Tech. Report 5546-93-9581, US Naval Research Laboratory, Washington, DC; Tech. Report Info-0474-1, Atomic Energy Control Board of Canada, Ontario, 1993.
3. D. Craigen, S. Gerhart, and T. Ralston, "An International Survey of Industrial Applications of Formal Methods, Volume 2 Case Studies," Tech. Report PB93-178564/AS, National Technical Information Service, Springfield, Va.; Tech. Report 5546-93-9582, US Naval Research Laboratory, Washington, DC; Tech. Report Info-0474-2, Atomic Energy Control Board of Canada, Ontario, 1993.
4. S. Gerhart, D. Craigen, and T. Ralston, "Observations on Industrial Applications of Formal Methods," *Proc 15th Intl. Conference on Software Engineering*, IEEE CS Press, Los Alamitos, Calif., 1993, pp. 24-33.
5. D. Craigen, S. Gerhart, and T. Ralston, "Formal Methods Reality Check: Industrial Usage," *Proc Formal Methods Europe*, Springer-Verlag, Berlin, 1993, pp. 250-268.
6. *The Procurement of Safety Critical Software in Defence Equipment (Part 1: Requirements, Part 2: Guidance)*, Interim Defence Standard 00-55, Issue 1, Ministry of Defence, Glasgow, Scotland, 1991.
7. C. Potts, "Software-Engineering Research Revisited," *IEEE Software*, Sept. 1993, pp. 19-28.



Susan Gerhart is director of the Research Institute for Computing and Information Systems at the University of Houston at Clear Lake, which administers a cooperative agreement with NASA's Johnson Space Center to provide a setting for partners from universities, government agencies, and industry to explore and apply new technologies. Her interests are using research and applying technology.

Gerhart received a BA in mathematics from Ohio Wesleyan University and a PhD in computer science from Carnegie Mellon University. She is a member of the IEEE Computer Society.



Dan Craigen is Director, Ottawa Operations, of ORA Canada. His research interests are the development and application of formal-methods technology, in particular the m-EVES (Environment for Verifying and Evaluating Software) and EVES verification systems.

Craigen received a BSc with honors and an MSc in mathematics from Carleton University. He is a member of the ACM and an affiliate member of the IEEE.

Ted Ralston is president of Ralston Research Associates, a consultancy for software engineering and European technology analysis. Ralston received a BSc in zoology and a BA in history from the University of Washington and an MS in history and languages from Oxford University.

Address questions about this article to Gerhart at RICIS, 2700 Bay Area Blvd., Houston, TX 77058-1098; gerhart@cl.uh.edu; Craigen at ORA Canada, 265 Carling Ave., Ste. 506, Ottawa, Ont., K1S 2E1, Canada; dan@ora.on.ca; and Ralston at p00833@psilink.com.