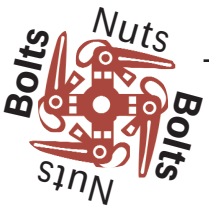


Requirements management has a critical effect on an organization's development costs and software quality. The authors have developed a method that allows incremental, systematic improvement of requirements engineering and builds on existing SPI models and standards, filling in the gaps to improve schedules, budgets, and product quality.

Capturing the Benefits of Requirements Engineering

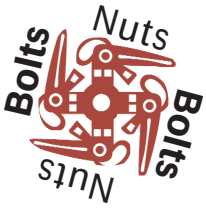
Pete Sawyer, Ian Sommerville, and Stephen Viller, Lancaster University

NUTS & BOLTS: Karl Wiegers and Dave Card, editors • kwiegers@acm.org / card@computer.org

Requirements problems are expensive and plague almost all systems and software development organizations. In most cases, the best you can hope for is to detect errors or omissions in the requirements in time to contain them before the product is released. With luck, nonessential functionality can be traded for product quality. All too often, however, the product is late, over budget, and of poor quality, while failing to meet crucial customer requirements.

Faced with recurrent requirements problems, organizations often turn to software process improvement for a solution.[1,2] Unfortunately, SPI seldom fixes problems rooted in the requirements process because existing SPI models concentrate on the downstream phases of development. The SEI's Capability Maturity Model for Software,[3] for example, explicitly addresses requirements processes only in the level two Requirements Management key practice area. The more recent Systems Engineering CMM[4] has greater coverage of requirements engineering, but is widely and incorrectly perceived to be aimed only at niche industries.

0740-7459/99/$10.00 © 1999 IEEE



Despite a half-century of progress in software development, many organizations continue to struggle with the elicitation, specification, and management of requirements. Many effective techniques are available to deal with these issues, but few are widely and routinely practiced by development organizations. This paper advocates complementing a sensible emphasis on existing requirements engineering "good practices" with other software process improvement initiatives. A requirements process maturity model provides a structure to assist organizations with the practical, phased adoption of improved requirements practices. This model recognizes that certain basic practices are both easy and cheap to adopt, while others must build on a solid foundation of systematic and institutionalized requirements practice to be successful. The dynamic and communication-intensive nature of software requirements engineering guarantees that a silver-bullet solution does not exist. However, the application of established good practices can help any organization improve the quality of its requirements, and hence the quality of its products.

—Karl Wiegers and David Card, Nuts & Bolts Editors

SPI has improved the process of developing products from requirements,[5] but it hasn't helped the development of requirements from customers. Consequently, many SPI programs stumble because they do not adequately address the requirements problems that underlie poor product quality. Timeliness, cost control, and quality may improve, but the level of achievable improvement is capped by flaws in the requirements process.[6]

The limitations of SPI motivated the Esprit Requirements Engineering Adaptation and Improvement for Safety and Dependability, or Reaims, project. The Reaims industrial partners develop mission-critical systems, where the quality implications of poor requirements handling are particularly serious. When Reaims started, changes in the partners' business sectors were forcing changes to their business processes. For example, an evolutionary shift from customer- to market-driven projects was forcing them to reexamine how they balance technical user and strategic organizational requirements. They needed practical measures to improve their requirements handling and to help them adapt.

Reaims has developed several novel requirements engineering methods and tools to address specific problems identified by the industrial partners. For example, the Reaims PREview method[7] helps structure the discovery of requirements and requirements sources. What distinguishes Reaims, however, is that we also address the problem of integrating new tools and methods into an organization. New techniques impose additional requirements on the requirements process; they can't simply be plugged in and expected to work. If appropriate measures aren't taken, a process won't be able to support or exploit a new method. The importance of the integration problem is often underrecognized and partly explains the poor industrial take-up[8] of new requirements methods.

We sought to lay a foundation of good practice to underpin the Reaims methods and tools. We also realized that industry's current enthusiasm for SPI provided an opportunity to stimulate a general improvement in requirements engineering practice. Our answer was to develop the Requirements Engineering Good Practice Guide (REGPG),[9] which draws on existing SPI models to define a process improvement framework for the systematic, incremental adoption of good requirements practice.

The REGPG reflects our conviction that, although many unsolved problems exist in requirements engineering, many more can be solved using well-established good practice. Even where problems are inadequately understood, the consequences for individual projects can usually be contained if adequate support exists within the requirements process.

The Requirements Process Maturity Model

The REGPG is based on an improvement framework that, like the CMM, uses multiple process maturity levels (see Figure 1). Maturity levels help characterize a process and set out a strategy for its improvement. Rather than requiring that a process be modeled in detail (a difficult and expensive activity), we can devise a checklist of questions that allows us to broadly classify a process as a level *n* process. We can then plan improvement to level *n+1* by knowing approximately what the process's existing capability is and what realistic improvement targets we can achieve from a set of possible improvement measures.

In contrast to the CMM, the REGPG has only three maturity levels. The CMM uses a five-layer framework, but the current state of the practice makes it doubtful whether any requirements processes exist that could be characterized beyond level 3, Defined.

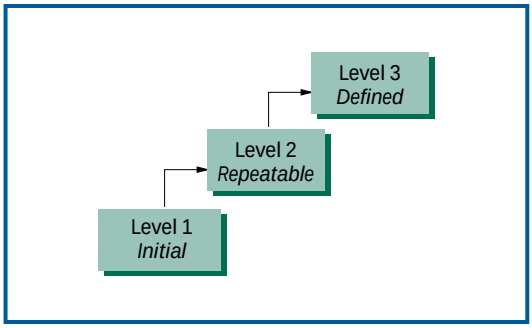


Figure 1. The three-level Reaims process maturity model.

We know of none from which we could extract generic principles, so we have limited our model to three levels.

♦ *Level 1*: Initial-level organizations have an ad hoc requirements process. They find it hard to estimate and control costs as requirements have to be reworked and customers report poor satisfaction. The processes are not supported by planning and review procedures or documentation standards but are dependent on the skills and experience of the individuals who enact them.

♦ *Level 2*: Repeatable-level organizations have defined standards for requirements documents and have introduced policies and procedures for requirements management. They may use tools and methods. Their documents are more likely to be consistently high in quality and to be produced on schedule.

♦ *Level 3:* Defined-level organizations have a defined process model based on good practices and defined methods. They have an active process improvement program in place and can make objective assessments of the value of new methods and techniques.

In our experience, nearly all requirements processes are at level 1. Most organizations have pockets of good practice, but their benefits are usually diluted by weaknesses elsewhere. The key to improvement is increasing the use of appropriate good practice. This is not simply a matter of buying more tools or converting to new approaches, such as object-oriented analysis, but of introducing the right practices, in the right order, at the right pace, and with the required degree of strategic commitment.

Faced with a large range of possible technical fixes and managerial procedures, an organization may find it difficult to decide where to start. The REGPG addresses this by distilling guidance on good practice into guidelines. These are designed to help organizations make a rational assessment of which practices offer the best cost and benefit tradeoffs for their practical needs.

Good Practice Guidelines

The REGPG describes 66 good practices that we abstracted from existing standards,[10] reports of requirements practices,[11,12] and the experience of Reaims partners. We have recognized that, while consensus exists about the utility of some practices, the value of others are project-, organization-, or application-domain-dependent. Similarly, some practices must be underpinned by other measures or require specialist expertise. To reflect this, we classify the practices according to whether they are basic, intermediate, or advanced.

♦ Basic practices represent fundamental measures that underpin a repeatable process. They are seldom technical solutions, but are usually focused on defining organizational procedures or documentation standards. Organizations should adopt basic practices before considering others.

♦ Intermediate practices are more technical and usually require that basic practices be in place for effective management. They help make the process systematic by, for example, using defined methods in conceptual modeling.

♦ Advanced practices require substantial specialist expertise and support continuous improvement. They also include practices that most benefit specialist domains. They must usually be underpinned by basic and, occasionally, intermediate practices.

Clearly, the classification we chose for some of the practices could be debated. The requirements engineering research community tends to underestimate the difficulty of introducing seemingly straightforward practices into a live process. For example, we think the collection of requirements from multiple perspectives or viewpoints is an apparently simple and commonplace practice. However, we classify it as an intermediate rather than a basic practice because it is costly to introduce the explicit management of multiple viewpoints. We have tried to err on the side of caution and concentrate on establishing a sound baseline for improvement.

It is not enough to rate practices as basic, intermediate, or advanced. In most cases, improvement must be achieved in carefully planned and moni-

Example Guideline

Defining a checklist or checklists will help focus the attention of requirements validators on critical attributes of the requirements document. These checklists can identify what readers should look for when they validate the system requirements.

Key Benefits:	Help focus the validation process
Costs of introduction:	Low to moderate
Costs of application:	Low
Guideline type:	Basic

Benefits

♦ Checklists add structure to the validation process, making it less likely that readers will forget to check aspects of the requirements document.

♦ Checklists help train people new to requirements validation. This is particularly important for customer management and end users who may not have requirements validation experience.

Implementation

These checklists focus on individual requirements; validation checklists should also be concerned with the quality properties of the requirements document as a whole and with the relationships between individual requirements. This can't be checked during requirements analysis as the requirements document is unfinished at that stage.

Questions that might be included in such a checklist should cover the following general issues:

1. Are the requirements complete? Does the checker know of any missing requirements, or is any information missing from individual requirement descriptions?
2. Are the requirements consistent? Do the descriptions of different requirements include contradictions?
3. Are the requirements comprehensible? Can readers of the document understand what they mean?
4. Are the requirements ambiguous? Are different interpretations possible?
5. Is the requirements document structured? Are related requirements grouped? Would an alternative structure be easier to understand?
6. Are the requirements traceable? Are requirements unambiguously identified with links to related requirements and to the reasons these requirements have been included?
7. Do the requirements document and individual requirements conform to defined standards?

Checklists should be expressed in a general way and should be understandable by people who are not system experts, such as end users. As a general rule, checklists should not be too long, 10 items or less. If you have more than this, checkers can't remember all the items and must continually consult the checklist. The danger is that the checklist will become too vague and that it is impossible to answer the checklist questions in any useful way. You must therefore find the right balance between generality and detail. Unlike program inspections, low-level checklists concerned with very specific faults are not good for requirements inspections because the requirements for different types of systems vary.

Checklists can be distributed and used to remind people what to look for when reading the requirements document. Alternatively, they can be used to indicate when a checklist item has been considered. This may be done with a simple database or spreadsheet.

Costs and Problems

This is not an expensive guideline to implement if you use a general checklist with questions like those listed above. To introduce the guideline, draw up an initial checklist based on the experience of people who have been involved in requirements validation.

If a checklist is simply used as a memory aid, there are no costs involved in applying the guideline. If the requirements checkers must mark each requirement against the checklists, some additional time is required, but not more than one or two minutes per requirement.

In principle, you should have few problems applying the guideline if you have a flexible process that allows people to ignore inappropriate checklist entries. Some requirements analysts may resent the introduction of checklists, but you should emphasize that checklists are designed to help them and that they must use their professional judgement.

tored stages rather than by dramatic, wholesale change. An organization aiming to attain level 2 will not introduce all the basic practices at once—that could fatally destabilize its process. It will instead introduce improvements incrementally, and will require an assessment of the costs and benefits. To help with this, the REGPG presents good practices in the form of guidelines (see the boxed text, "Example Guideline"). Each guideline provides a qualitative assessment of the following factors:

♦ The key benefits of the practice. This outlines the improvements that can be expected by adopting the practice.

♦ The cost of introducing the practice. This indicates the level of effort and investment needed to integrate the practice in an existing process, such as staff training and support systems. We distinguish between the costs of introduction and application because even if the eventual benefits are large, the introduction of some practices needs careful timing.

Table 1
Requirements Management Good Practices

Good Practice	Cost of Introduction	Cost of Application	Guideline Classification	Key Benefit
Uniquely identify each requirement	Very low	Very low	Basic	Provides unambiguous references to specific requirements
Define policies for requirements management	Moderate	Low	Basic	Provides guidance for all involved in requirements management
Define traceability policies	Moderate	Moderate to high	Basic to intermediate	Maintains consistent, traceable information
Maintain a traceability manual	Low	Moderate to high	Basic	Records all project-specific traceability information
Use a database to manage requirements	Moderate to high	Moderate	Intermediate	Makes it easier to manage large numbers of requirements
Define change management policies	Moderate to high	Low to moderate	Intermediate	Provides a framework for systematically assessing change
Identify global system requirements	Low	Low	Intermediate	Finds requirements likely to be most expensive to change
Identify volatile requirements	Low	Low	Advanced	Simplifies requirements change management
Record rejected requirements	Low	Low	Advanced	Saves re-analysis when rejected requirements are proposed again

For example, it may be impractical in the short term to send staff for training close to a project milestone.

♦ The cost of applying the practice. This indicates the effort required to use the practice effectively once introduced. This helps an organization do a cost–benefit analysis of its process improvement measures since some practices consume additional resources.

The following guidelines are organized according to the process deliverables or activities to which they mainly contribute.

♦ The requirements document: Structuring and organizing the requirements document to effectively communicate requirements to customers, managers, and developers.

♦ Requirements elicitation: Acquisition of requirements and constraints from system stakeholders, the application domain, and the system's operational and organizational environments.

♦ Requirements analysis and negotiation: Identification and resolution of issues arising from the elicited requirements.

♦ Describing requirements: Writing requirements to aid readers' understanding.

♦ System modeling: Developing conceptual models to aid understanding and analysis of the requirements and their implications for the proposed system.

♦ Requirements validation: Establishing procedures to check for correctness, completeness, consistency, and compatibility. Ensuring that requirements are verifiable and that quality standards are adhered to.

♦ Requirements management: Managing requirements information throughout the development life cycle.

♦ Requirements engineering for critical systems: Practices for handling requirements for critical systems, such as safety or reliability requirements.

As an example, Table 1 lists the guidelines for requirements management. The association of practices with process deliverables or activities offered us a choice of model "architecture":

1. We could adopt a staged architecture like the CMM and associate each process deliverable or activity with a particular maturity level. For example, to reach level 2, an organization should adopt all the good practices in the requirements document describing requirements and requirements management.

2. We could adopt a looser, continuous architecture, like SPICE,[13] where improvement can be achieved by adopting practices across a range of process deliverables or activities.

We selected option 2 partly because of its flexibility. More importantly, however, we could not say

which process deliverables or activities should take priority over others in all cases. We have a more vague view of a generic requirements process than CMM's designers had for the software development process. In practice, it often makes sense to concentrate resources on a particularly weak process deliverable or activity, but where weaknesses exist across the requirements process, the prioritizing of improvements needs to be more flexible.

Assessing Processes and Planning Improvements

A key requirement of any process improvement model is usability. This involves determining the baseline from which improvement must begin, which means the process engineer has to discover the existing strengths and weaknesses. This can be surprisingly difficult because people inevitably have their own perspectives on a process. Process actors will bias their model and external process engineers must collect and rationalize each person's perspectives into a coherent model. In other words, modeling a process is like requirements engineering.

When process improvement is certified, this is a difficult issue. CMM-accredited process assessors take several days to inspect an organization and establish its position in the maturity level framework. This provides a degree of objectivity and ensures that all organizations know in advance the criteria by which they will be judged. Improvement planning is based on addressing the key practices in the key process areas required to attain the next maturity level.

The REGPG is not intended for certification, so we can afford to be less rigorous. However, the advantages of an assessment scheme, like CMM, that uses a combination of data collection and analysis are compelling. We recommend a hybrid that supplements checklists with the knowledge of the process actors and the judgment of the person performing the assessment. This does not build a detailed model of the process, but does reveal what practices are in use and to what extent. This allows us to position the process within the improvement framework and identify where the use of good practice is weak. In a large organization, the extent of good practice use will vary according to project, engineer, and customer. To accommodate this inevitable variation, each good practice is assessed by these criteria:

1. Standardized (score 3). The practice has a documented standard in the organization and is followed and checked as part of a quality management process.
2. Normal use (score 2). The practice is widely followed in the organization but is not mandatory.
3. Discretionary (score 1). Some project managers may have introduced the practice, but it is not universally used.
4. Never (score 0). The practice is never or rarely applied.

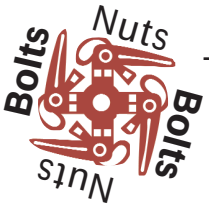
The assessment process includes phases for selecting people to interview, initial scoring, refinement, and final maturity level calculation. This permits resolving uncertainties by renegotiating the assessor's model of the process with those of the actors. The maturity level is calculated by summing the numerical scores for each practice used (see Table 2).

Table 2
Assessment Scores and Maturity

Maturity Level	Assessment Score
Initial	Less than 55 in the basic guidelines
Repeatable	Above 55 in the basic guidelines but less than 40 in the intermediate and advanced guidelines
Defined	More than 85 in the basic guidelines and more than 40 in the intermediate and advanced guidelines

A continuous-improvement architecture must classify the maturity of a process without being over-prescriptive about the practices at each level. Our approach, like SPICE, measures the extent to which individual practices have been implemented and institutionalized to assess the maturity level of each process area, because this allows process engineers to be flexible when targeting improvement efforts. Our experience indicates that, while this flexibility is welcomed at the operational level, a management requirement frequently persists for a measure of the overall process maturity. The overall REGPG maturity assessment is designed to satisfy this requirement. It serves as a rough indicator of how mature a requirements process is in the context of wider SPI programs.

The scheme is based on a combination of empirical data, analysis of common practice, and what we know about the cost and utility of the individual practices. It was formed by the Reaims industrial partners' experience of addressing requirements process weaknesses. Their cumulative experiences show the requirements practices that are consistent with each maturity level. We used this experience



to refine the classification scheme into reasonable, ball-park figures for most scenarios. It shows the broad characteristics of processes at each level.

There are 36 basic practices. In a typical repeatable process, most of the basic practices defining organizational procedures or documentation standards are standardized, with many of the remaining basic practices in normal or discretionary use. A defined process, by contrast, requires more systematic support provided by the 21 intermediate practices. A typical organization with a defined process might have standardized appropriate modeling methods and requirements management tools. It will also be capable of selecting and applying new intermediate practices and even some advanced practices.

The REGPG seeks to initiate the consolidation of industrial requirements engineering practices rather than to prescribe a standard.

Where requirements handling problems are evident, the improvement exercise will seek to address them. Problems are not always evident, however, and it may take some detective work to trace product problems back to the requirements process. Regardless of the problems identified, the improvement goals must be realistic. It won't be possible to completely eliminate requirements rework, but it is reasonable to aim for a 20 percent reduction in the number of reworked requirements in each improvement cycle. Achieving this requires establishing causality between the evident problems and the weaknesses discovered in the process. The improvement effort should focus on the poorly supported process activities that are suspected of contributing to the problems. The good practice guidelines are designed to help select and prioritize which good practices to adopt.

Process improvement is not a one-step process. Even addressing the most urgent process problems may take several improvement cycles since good practice must be introduced incrementally and its performance verified. New practices will need bedding in, particularly when staff must be trained or long-established working practice must be changed. The operation of the new practices should be monitored. Feedback on how well they work should be collected to verify expectations, to assess the impact on the improvement goals, and to help tune the process for efficiency.

Scoping Future Practices

The REGPG is not intended to provide revolutionary solutions to requirements engineering. We don't think that practical revolutionary solutions exist. However, there is a body of wisdom locked up in existing standards, organizational experience, and applied research work that hasn't been disseminated in a form easy to evaluate or adopt. Reaims released REGPG as a first step toward raising the state of the practice.

At the British Computer Society Requirements Engineering Special Interest Group meeting at the University of York on 4 February 1998, Andy Vickers observed that while most researchers view requirements engineering as a problem of product complexity, the most pressing problems faced by industrial practitioners arise from organizational complexity, which is fundamentally a process issue. If the requirements process cannot cope with changing requirements or large volumes of documentation, then projects will fail. We believe that the basic practices that form the mainstay of progression from initial to repeatable requirements processes offer the greatest leverage on organizational complexity.

Although the REGPG includes a CMM-like improvement framework, it is not intended for accreditation. Industry's enthusiasm for SPI, however, suggests a growing trend for internally driven SPI programs where accreditation is not the goal. We tried to exploit this by adopting an improvement framework that helps orient requirements process improvement with other SPI initiatives. The current paucity of accepted standards in requirements engineering led us to design our improvement model to accommodate a wide spectrum of views and practices. Hence, the REGPG seeks to initiate the consolidation of industrial requirements engineering practices rather than to prescribe a standard.

REGPG has had a number of different effects on the Reaims industrial partners' organizations, and other organizations through dissemination by the Reaims partners. It met its original goal of complementing their SPI programs by, for example, helping to meet requirements management prerequisites that are only implicit in the CMM level 2 KPA. It has also helped lay the foundation for the integration of the methods and tools developed by

Reaims. Its most significant effect, however, has been to raise management awareness of the importance of requirements engineering to an organization's strategic cost and quality goals.

Few organizations can afford to radically change their existing requirements processes. The advantage of the REGPG is that it allows incremental improvement by matching the most effective measures with the most pressing problems. ❖

Acknowledgments

We would like to thank all the Reaims partners: GEC-Alsthom Transport, Adelard, Aerospaciale Avions, Aerospaciale Protection Systems (APSYS), Digilog, TÜVit, and the University of Manchester. We are also grateful to the reviewers, whose comments have improved this article.

References

1. W. Humphrey, Managing the Software Process, Addison Wesley Longman, Reading, Mass.,1989.
2. S. Zahran, Software Process Improvement: Practical Guidelines for Business Success, Addison Wesley Longman, Reading, Mass.,1998.
3. M. Paulk et al., Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-24, Software Eng. Inst., Pittsburgh, Pa., 1993.
4. R. Bate et al., A Systems Engineering Capability Maturity Model, Version 1.1, CMU/SEI-95-MM-03, Software Eng. Inst., Pittsburgh, Pa., 841993.
5. M. Diaz and J. Sligo, "How Software Process Improvement Helped Motorola," IEEE Software, Sept./Oct. 1997, pp. 89-96.
6. A. Hutchings and S. Knox, "Creating Products Customers Demand," Comm. ACM, Vol. 38, No. 5, 1995, pp. 72-80.
7. I. Sommerville et al., "Viewpoints for Requirements Elicitation: A Practical Approach," Proc. 3rd Int'l Conf. Requirements Eng., IEEE Computer Soc. Press, Los Alamitos, Calif., 1998, pp. 74-81.
8. P. Morris et al., "Requirements Engineering and Industrial Uptake," Proc. 3rd Int'l Conf. Requirements Eng., IEEE Computer Soc. Press, Los Alamitos, Calif., 1998, pp. 130-137.
9. I. Sommerville and P. Sawyer, Requirements Engineering—A Good Practice Guide, John Wiley & Sons, New York, 1997.
10. C. Mazza et al., Software Engineering Standards, Prentice Hall, Upper Saddle River, N.J., 1994.
11. M. Lubars et al., "A Review of the State of the Practice in Requirements Modeling," Proc. IEEE Int'l Symp. Requirements Eng., IEEE Computer Soc. Press, Los Alamitos, Calif., 1993, pp. 2-14.
12. K. El Eman and N. Madhavji, "Measuring the Success of Requirements Engineering Processes," Proc. 2nd IEEE Int'l Symp. Requirements Eng., IEEE Computer Soc. Press, Los Alamitos, Calif., 1995, pp. 204-211.
13. M. Konrad and M. Paulk, "An Overview of SPICE's Model for Process Management," Proc. 5th Int'l Conf. Software Quality, Am. Soc. for Quality, Milwaukee, Wisc., 1995, pp. 291-301.

About the Authors

Pete Sawyer is a senior lecturer in the Computing Department at Lancaster University, where he has held a variety of positions since 1986. He is currently spending six months working at the Institute for Systems, Informatics, and Safety at the European Commission's Joint Research Centre in Ispra, Italy. His principal research interests are requirements engineering, dependable systems, and software process improvement.

Sawyer holds a BSc and PhD in computer science from Lancaster University. He is a member of the IEEE Computer Society and the ACM.

Ian Sommerville is professor of computer science and chair of the Computing Department at Lancaster University. He has been involved in software engineering research and teaching for the past 20 years and his current research interests include requirements engineering, system evolution, and human and social factors in systems design.

Sommerville received a BSc in physics from Strathclyde University and an MSc and PhD from St. Andrews University. He is a member of the ACM, the IEEE Computer Society, and the British Computer Society, and is a fellow of the IEE.

Stephen Viller is a research fellow in the Computing Department at Lancaster University. His research interests are in the fields of requirements engineering and computer-supported cooperative work, focusing on improving support for the design of cooperative systems. He is currently completing his PhD on a human-sciences-informed process improvement method directed at the requirements process for safety-critical systems design.

Viller has a BSc in computation from the University of Manchester Institute of Science and Technology and an MSc in cognitive science from the University of Manchester. He is a member of the IEEE Computer Society and the ACM.

Address questions about this article to Sawyer at the Computing Department, Lancaster University, Lancaster, UK, LA1 4YR; e-mail sawyer@comp.lancs.ac.uk.