A Requirements Engineering Process for OPEN Development

Danielle Fowler
Swinburne University of Technology
Email: dfowler@swin.edu.au

Brian Henderson-Sellers
University of Technology, Sydney
Email: brian@socs.uts.edu.au

Paul Swatman
Deakin University
Email: paul.swatman@deakin.edu.au

Abstract

The impact of forces such as the internet and the rise of e-commerce has led to an increasingly distributed and diverse style of systems development environment. With this change has come the call for methods suited to this type of development. The object technology community has developed a number of techniques which facilitate development in such an environment (e.g. components, design patterns and templates). However comprehensive software process support is still lacking, particularly in the area of requirements elicitation and specification. This paper describes an implementation of the OPEN methodology which incorporates formal specification into the requirements engineering process through integration of the requirements engineering method FOOM, and how such an approach benefits the types of systems development projects described above.

Keywords

Requirements engineering, formal specification, object orientation, methodology, FOOM, OPEN.

THE TARGET SYSTEM DEVELOPMENT ENVIRONMENT

Brooks (1994) argued that building software is hard, and will continue to be, because of four inherent properties of modern software systems: complexity, conformity, changeability and invisibility. Addressing complexity as an issue, systems development methods must take into account

- the complexity of the problem domain
- the difficulty of managing the developmental process
- the flexibility possible through software, and
- the problems of characterising the behaviour of discrete systems (Booch 1994, p.5)

This complexity has only increased in recent years, as system development projects are increasing developed, owned and run across a variety of platforms, geographical locations and organisational areas. Examples are prevalent: electronic commerce initiatives, for instance, range from supply chain improvement schemes such as evaluated receipt management (ERS) or quick response (QR), to internet-based catalogue or payment systems. All are dependent on IT, all involve the cooperation of more than one organisation, will likely require the technical and management skills of people from multiple organisations, and many are distributed across platform, distance and time zones. The current growth in open-source projects, as another

example, presents challenges for the way traditional project management is approached (Lin and Henderson-Sellers 1998), as little conventional software engineering practice applies to systems developed in this manner.

In the light of these challenges, where traditional views on the requirements of a software engineering process do not hold, or are insufficient, development methods must either be aimed at specific domains and development types ("strong" vs "weak" methodologies (Vessey and Glass 1998)) or must be flexible and tailorable to a wide range of situations.

Current efforts in the Object Technology community to address this increasingly diverse environment have focused on a component, or "plug and play" based view of development: the construction of common models suited to particular environments (e.g. the business objects of a financial institution, or design templates useful in the building of accounting systems) which allow not only for reuse but also, hopefully, increased quality through peer review/refinement. Developments in the areas of OO enterprise modelling and architecture are similarly focused on leveraging the strength of the object paradigm in facilitating the partitioning of complex situations into manageable parts, and allowing for reuse of not only code, but architecture, design and even analysis level models. With this change in approach comes the need for process support, and flexibility in object-oriented methodologies.

THE COMBINING OF METHODS

FOOM (Formal Object Oriented Method) (Fowler 1996; Swatman 1992,1996) is the result of an ongoing research project which originated at Curtin University in the early 1990s with the focus of improving the understanding and quality of the requirements elicitation and specification process, and particularly in examining the benefits of using formal specification in MIS systems.

The core theoretic principles on which FOOM is based were socio-organisational theory, particularly SSM, object orientation, and formal methods. The development of FOOM has evolved over the course of a number of separate studies, the result of which has been the development of an IS development approach grounded in these three areas:

socio-organisational contextual analysis, following the work of Checkland (1981, 1989, 1995) and Checkland and Scholes (1990) which, in the general case, denies the existence of a single, objective requirements specification waiting to be discovered by the systems analyst

mathematically formal specification languages (Fraser et al. 1994) in particular, the object-oriented specification language Object-Z (Duke et al. 1991) by means of which the abstract characteristics of classes may be described precisely and unambiguously

the object-oriented approach (Booch 1994; Rumbaugh et al. 1991; Henderson-Sellers and Edwards 1994) to systems modelling.

All three areas offer a contribution to the systems analysis and requirements modelling process; in combination, they offer the potential for both increasing understanding of the problem situation and representing it in a precise and rigorous manner.

Why Formal Specification?

Formal specification techniques offer a solution to the problem of poor requirements specification by enabling a statement of requirements to be specified in an unambiguous, precise manner which can be reasoned about formally. Such a document allows for distributed or outsourced development of the requirements specification, and enables improved understanding and communication of requirements between users and developers.

Traditionally, the industrial uptake of formal methods (Hinchey and Bowen 1995; Craigen et al. 1993) has been most noticeable in safety and security-critical systems (Bowen and Stavridou 1993; Gerhart et al. 1994) where the cost of software failure could be catastrophic. More recently, however, formal methods have gained popularity in those systems which must achieve commercial viability (Gerhart et al. 1994): mission critical systems, where quality is paramount and the perceived increased cost (largely a myth (Bowen and Hinchey 1995)) and resource demands associated with using formal methods are no longer automatically considered insurmountable barriers.

In particular, it has become increasingly recognised that formally specifying systems in the IS domain has tangible benefits where elicitation of functional requirements is much more open to misinterpretation than in the relatively more "objective" scientific/technical context. Perhaps the most important benefit in improving the requirements specification process is the contribution they can make to better problem understanding (Hall 1990; Swatman and Swatman 1992).

The precision of formal methods offers the potential for enhanced communication and validation of the requirements specification process. Not only can specifications be communicated with more precision downstream along the development channel (improving the process of producing products from specifications), but more precise validation of the specification against the users' needs is possible, improving the quality of requirements captured from the customer.

The traditional benefits of using formal methods include:

- higher quality systems - the match between computer programs and their specifications can be mathematically verified
- a more manageable development process - it is simple to compare a formal requirements specification with the progress which has been made in implementing the system
- cheaper, faster development - although the requirements specification phase may be longer, overall implementation time and cost are reduced due to the early identification of specification errors, omissions or ambiguities (Cunningham et al. 1985; Hall 1990).

As their potential for MIS development has become established, several research projects have been initiated to examine the incorporation or integration of formal specification techniques into mainstream IS methodologies (e.g. the SAZ project (Polack and Mander 1994; Mander and Polack 1995), which is looking at the integration of the formal specification language Z (Spivey 1992) and the SSADM methodology). These approaches have all tended to focus on combining formal specification with traditional structured analysis or design, however, and on using the formal specification process to yield the traditional, "downstream" benefits: in other words, as a way of facilitating greater understanding of the specification by developers. They have not focused on the promise of formal specifications as a means of improving the requirements validation process.

The additional benefits to be gained from using formal methods in the requirements engineering process for the IS community include:

- more relevant systems - the use of formal methods as an analytic tool assists in identifying conflicts among the various *Weltanschauungen* (world views) that will exist regarding the functional goals of a system
- providing a deeper insight into the system being specified
- a formally documented requirements specification, which provides a precise medium for communication, both between specifier and developers and between specifier and clients
- formal specifications, which provide a clear basis for acceptance testing, and also for litigation in the event of legal disputes

FOOM was developed as a requirements engineering approach incorporating the use of formal specification techniques within the requirements specification process in order to take advantage of these benefits.

The Benefits of Object Orientation

The second pillar of FOOM's development is object orientation. Much more widely adopted than formal methods, object orientation has become an accepted approach within the information systems community. We were interested in OO because object technology offers a useful way of thinking about software based on abstractions that exist in the real world (Henderson-Sellers and Edwards 1994; Shlaer and Mellor 1992). The key concepts characterising object-orientation, including precision, abstraction, encapsulation, modularity, relationship hierarchies, polymorphism and so on offer benefits in the modelling process. Jacobson et al. (1995) argue that the concepts underlying object modelling allow object models to be comprehensive, understandable, changeable, adaptable, and reusable:

- comprehensive - because it is possible to break down classes hierarchically, allowing an understandable overall picture of the business being modelled
- understandable - the business is described in terms of objects, which often have a direct link to occurrences in the real world. This supposedly smaller semantic gap between OO models and user perceptions of the real world is often put forward as one of the most valuable benefits of OO (Henderson-Sellers and Edwards 1994; Booch 1994) although this does not necessarily equate to easier validation of OO models
- changeable - changes usually relate to a single class, and can therefore be introduced without affecting other parts of the model
- adaptable - it is possible to specialise existing classes via inheritance, by inserting the adaptation into classes that are specialisations of more abstract classes
- reuseability - another widely promoted benefit. Classes can be built and handled as components, like Lego™, ready for use in other systems.

The primary benefit of the object-oriented approach relevant to more effective specification of systems, which was our goal, can be summarised as facilitating better communication, not only between developers but also between developers/specifiers and clients (the idea of the natural modelling paradigm). It offers a facility for partitioning and handling complexity in system models, thereby improving the validation process by increasing understanding of the systems model and hence increasing the likelihood of a quality product. This facility has been borne out in the predominant use of OO modelling and languages in environments characterised by some kind of distributed development (users, developers, platforms) such as internet systems.

We chose to use an OO formal specification language because of the synergy between formal methods and object orientation. Formal methods support object orientation by:

- improving the theoretical foundation of object-oriented methods. The solutions to a number of problems associated with object-oriented methods have been based on heuristics, rather than on firm theoretical ground. An example is the interference of inheritance with the class interface. Formal methods can inject a firmer foundation to the object-oriented methods by validating the soundness of these heuristic solutions through formal reasoning
- developing better specifications for objects. The object-oriented approach involves building system components which are connected by their interfaces. Formal specifications offer unambiguous and precise definitions for these interfaces
- proving the correctness of critical components. For components which are used frequently, formal methods can be used to ensure correctness. Bugs or problems found in

frequently reused components can have serious repercussions: changes made may require many programs to be recompiled and tested to ensure they still work correctly.

Object orientation, in turn, has benefits to offer to the further development of formal techniques. Although formal methods have been proposed as a mechanism for steering development towards a more engineering-like discipline, they lack support for software engineering practices such as teamwork, maintenance and reuse. Object orientation offers structuring concepts such as a library mechanism, modules and naming conventions to formal methods, and can also promote the reuse of specifications (Lano and Haughton 1994).

Subjectivity in the Requirements Engineering Process

An important challenge in incorporating formal techniques into the IS domain is combining the rigour of mathematical methods with more intuitive semi-formal notations, in order to enhance software quality but also to appeal to users participating in the specification development. The rationale and issues associated with using formal specification techniques and object technology have already been presented. There is an area of concern within requirements engineering which they do not address, however. Possibly the area of greatest difficulty in developing systems is overcoming the problems with communication: the "specification problem" (both of elicitation and validation).

The effectiveness of the requirements elicitation and specification process directly affects the chances that an intervention into the situation taken will be appropriate. Historically, most system development methodologies have begun with a requirements analysis phase, carrying the implicit assumption that the system to be built (and it is almost always assumed a computerised system or artefact will need to be constructed) is well understood; that is, the problem context is well understood. This is Checkland's (1981) "hard" systems approach, with its underlying positivist, objectivist viewpoint. It has long been the traditional approach of the computer science and software engineering communities. A single, objective statement of the requirements of the system (for everyone) is assumed to exist, waiting to be uncovered.

Although the information systems community in the past has also often taken this approach (what Winograd and Flores (1986) call the "rationalist tradition") to a large extent (Orlikowski and Baroudi 1991), increasingly attention has focused on an alternative "soft," or subjective approach to the problem. The literature has characterised the dichotomy in approaches in several ways: "hard" and "soft" by Checkland (1981); functionalist vs interpretivist by Burrell and Morgan (1979); rationalistic vs hermeneutic by Winograd and Flores (1986). Soft approaches to IS development have emerged, focusing on understanding the organisational context - on working out the nature of the problem to be addressed (which may then involve the construction of a computerised system). This approach has gained favour over recent years as the IS discipline seeks to understand more completely the domain within which its practitioners work. Rather than accepting that an objective reality exists which can be modelled, the soft tradition in IS focuses instead on producing models which correspond to the conceptual models of the system of the users involved. This tradition considers "reality" to be a subjective notion; an organisation is comprised, then, not of numerous people dealing with systems of objective data, but information systems which have meanings corresponding to the various viewpoints of the people involved.

The strength of these socio-organisational or socio-technical approaches to development is their focus on the system context, increasing the likelihood that the impact of intervention within the organisational context will be understood, and thus that any intervention will be appropriate. One of the best known examples of these "soft" methodologies is the Soft

Systems Methodology (SSM). Checkland (1995) summarises four key thoughts which have dictated the emerging form of the approach:

- *purposeful activity* All problem situations have at least one thing in common: they all contain people - people who, in the face of ambiguity and conflict, are trying to take purposeful action. SSM takes a set of activities, linked in such a way as to constitute a purposeful whole as a kind of system, known as a human activity system. Techniques for modelling such systems (root definitions, CATWOE analysis, conceptual models) exist
- In recognising the subjectivity of experience, every model of a notional purposeful whole in SSM is built according to a declared worldview, or *Weltanschauung*
- *holons* From its start in systems engineering, SSM has moved the concept of systemicity from the view that "the world contains systems" (hard systems thinking) to the view that "learning about the world can be organised systemically" (soft systems thinking). Holons are abstract notions of purposeful wholes which may or may not map onto reality, but are considered relevant to debate about it. Models in SSM are only devices which enable the coherent exploration of perceptions of the real world to take place
- *activity models and information systems* A model of a purposeful system may be examined to answer questions about data and/or information support for the activity, and may therefore be linked to an associated set of data/information items required to enable the activity to occur. As information/data support is a feature of any real-world problem situation, information studies are common among uses of SSM.

The concepts and philosophical view of SSM form part of the theoretical basis for FOOM, although the actual notations and process are not necessarily incorporated. It is, rather, the ideas of Weltanschauung and the recognition of the need for conflict identification and resolution which have been incorporated.

Characteristics of the Environment of Use

The individual strengths of formal specification techniques and object orientation makes an approach combining these qualities particularly suited to an increasingly common systems development environment where the developers, clients, and/or users may be geographically distributed. The growth of inter-organisational systems or distributed systems is not new, but has certainly been aided by the internet as a communication mechanism.

The objective in developing FOOM was to investigate the development of a method applicable to commercial as opposed to scientific systems, but that remained too wide a focus. The FOOM research programme was directed toward the elicitation and specification of requirements in the more specific application domain of Inter-Organisational Systems (IOS) and Electronic Data Interchange (EDI).

There is now considerable interest in requirements engineering for global systems development efforts, particularly in inter-organisational or distributed development supported by the internet as a communication mechanism. In order for such development efforts to succeed an appropriate contextual and requirements analysis must first be undertaken.

As the increasing demand for electronic commerce emphasises the need for developing inter-organisational systems effectively, understanding the requirements of such systems is crucial. Multi-organisational information systems possess a number of characteristics which distinguish them from the "standard," characteristics which have a significant influence over the choice of IS development method employed:

- because of their scale and the numbers (and often wide geographic distribution) of interested parties, a prototypical approach to IS development is normally inappropriate for development. An appropriate IS development method for this context may focus on

requirements engineering as a discrete phase, and would benefit from tools and techniques offering precise, accurate communication

- in-house information systems development is, in general, not possible (if there is more than one organisation involved, each part of the development must be outsourced by at least one of them). Consequently, an appropriate requirements engineering method must provide support for outsourced development and for modularity and precise contractual arrangements
- it is unreasonable to assume the existence of clear non-conflicting goals within a single organisation - clearly this problem is exacerbated in the case of more organisations. An appropriate requirements engineering method must take account of the subjectivity of perception.

These characteristics introduce (or exacerbate) problems with systems development: difficulties associated with establishing clear lines of authority and responsibility across organisational boundaries; difficulties associated with establishing procedures which span organisational boundaries; difficulties caused by time delays associated with both inter- and intra- organisational communication.

FOOM addresses these problems, and provides several primary benefits with regard to the complexities of inter-organisational systems development. A formal specification offers a clear, unambiguous method of intra-organisational communication, for both users and developers. FOOM also allows a holistic view to be taken of the system - improving the inter-organisational communication between the various developers. In effect, it enables asynchronous communication between developers situated across a wide geographical area. Although FOOM was designed to be relevant to requirements engineering projects undertaken in an inter-organisational systems (IOS) domain, it is also relevant to other systems development projects where a prespecification based approach to requirements engineering is possible.

WIDER PROCESS CONSIDERATIONS: THE BENEFITS OF OPEN

The scope of the FOOM process covers both the early and late stages of requirements engineering, which are both often ignored in object-oriented methodologies, and seldom the focus of software process improvement models. Sawyer et al. (1999), for instance, point out that existing SPI models concentrate on downstream phases, with the CMM having poor coverage, although the newer Systems Engineering CMM has improved support (but the perception of niche applicability only). As a result, SPI programmes have traditionally helped transform requirements into products, but not the development of requirements from customers. FOOM is focused on both process stages, using formality for traditional strengths in specification precision, but also in validation with users as a way of better capturing requirements.

Originally an outline process model (Swatman 1992) which synthesised formal modelling, object orientation and the socio-organisational view into an approach to systems development, FOOM evolved over the course of a long action research study (Fowler 1996) aimed at refining the process/method in its intended setting: the commercial IS environment. Although the focus of this refinement was the requirements engineering or analysis phase, FOOM was designed to connect cleanly with any OO development methodology, or indeed any methodology with a discrete requirements analysis phase - a primary benefit of developing a formalised requirements specification document is that it provides a precise communication mechanism for developers, clients and users, and offers a clear basis for acceptance testing - qualities especially useful in outsourcing projects. Remaining within the OO paradigm,

however, allows for the benefits associated with reuse, as well as reducing the introduction of error in translating between modelling paradigms. In particular, FOOM was designed to be compatible first with the MOSES methodology (Henderson-Sellers and Edwards 1994) and now OPEN.

OPEN (Object-oriented Process, Environment and Notation) is a process focused, third generation, full lifecycle method(ology) (Henderson-Sellers et al. 1998). Initially created from the merger of the MOSES, SOMA and Firesmith methods, OPEN is essentially a framework for "third generation" OO development methods which has also incorporated concepts from BON, Martin/Odell, Discovery, UML and others. It provides a tailorable lifecycle, a metamodel level process description, techniques and representations (a modelling language). Individual methods conform to the OPEN methodological framework by adopting some or all of the framework specification: lifecycle, tasks, techniques and modelling language.

An OPEN process instantiation is made up of activities, which interact with each other via message passing. The "methods" of an activity are its tasks, which are carried out by people using techniques. A two dimensional matrix is used to cross link tasks with the techniques used to achieve them. In this way a tailored lifecycle process may be designed for a particular organisation, or application domain. The lifecycle model in OPEN is contract-driven, with activities (represented by objects) interacting only if the pre- and post- conditions of the contract between any pair of activities is met.

The preferred modelling language used within OPEN is OML (OPEN Modelling Language), which is a notation plus a metamodel, although other common modelling languages, such as the UML, can be used. OML object notations are used within the FOOM specification structure (the static OO diagrams used in tier three), and the various FOOM notations (event chain and communication diagrams: (Fowler et al. 1995; Wafula and Swatman 1996) have been designed to be visually consistent with first MOSES and now OML.

THE OPEN/FOOM REQUIREMENTS ENGINEERING PROCESS

The FOOM approach to requirements "engineering" is a cyclic one in which initially the requirements (the basis for intervention in the problem context) are undefined - indeed, the problem context itself may be ill- or un-defined. In cycles of Information Analysis, Modelling and Debate and Validation (see below) the situation and requirements of the required intervention are established, agreed and documented.

In outline, contextual data relating to the problem situation are transformed into information which is modelled and structured using a variety of complementary techniques which range from informal to formal in nature. These models are validated both against the contextual data and to ensure internal consistency. Debate ensues relating to:

- the various stakeholders' perception of the current and desired states of the problem situation
- elegance and problem-solving value of the architecture of the abstract model(s). Typically, the debate and validation activity provides material for further cycles of FOOM analysis. In the final cycle, of course, the agreed specification of the problem and the requirements for the "solution" form the output of the activity.

Activities Involved

The process begins with the "requirements" (the basis for intervention in the problem context) undefined - at this stage the problem context itself is not well defined. "Contextual data" about the problem situation are elicited and form the input to the requirements definition process, which is comprised of three iterative activities:

information analysis In this activity, we establish perspectives on the problem context and view the problem context from each of these perspectives. The information analyst's role is that of facilitator. The purpose of this activity is to transform data regarding the context (later the system) into meaningful information. Data are rendered meaningful to a person through an individual act of interpretation. Each individual participating in the process will have their own valid but inherently subjective view of reality; in this way the subjective perspectives on the "raw data" elicited are recognised. Information analysis is also concerned with the resolution of identified conflicts between the different worldviews involved. Over time, this process will lead to the refinement of a set of purposeful human activity systems, one of which will ultimately be formally specified

modelling This is an essentially technically-led activity in which problem statements regarding the problem situation are modelled. The system/contextual information produced by the information analysis phase is modelled, producing problem statements of the area of investigation. Informal, semi-formal and formal models of the problem domain and intervention requirements may be constructed. It is important to note that the different types of modelling, including the use of SSM techniques to define the problem situation within which intervention is being considered, occur in parallel, rather than sequentially. The various modelling processes each contribute to the understanding of the situation, and are interdependent

- the informal modelling techniques used are to a large extent discretionary, based upon their suitability to the problem context. If the user acceptor is familiar with Data Flow Diagrams, for instance, their use may be an effective way of validating requirements with some users. The use of some type of behavioural modelling approach, however, is a necessary additional requirement in this process. This type of model is needed to assist validate the complete, formal specification with the user-acceptors who are ultimately responsible for the sign-off of the specification, and who therefore need a more thorough understanding of the requirements specification. Behavioural models allow for "road maps" to be drawn through the OO model, allowing validation to centre around activites/process that the user is familiar with. Event chain diagrams (Fowler et al. 1995) were specifically designed for this purpose, but other diagrams may be used if suitable. We have found that use cases, although they are popular with the OT community as requirements models, are not suitable for this purpose. Use cases are not object-oriented, and do not provide a way of presenting the behaviour of all parts of the system (as opposed to only the interfaces with the actors in the system) for validation (see Korson 1998; Firesmith 1995, for discussions on the limitations of use cases).
- semi-formal modelling techniques include conventional static and dynamic OO diagrammatic techniques, and the communications modelling notation developed by Wafula (Wafula and Swatman 1996). As with behavioural modelling diagrams, these form an integral part of the requirements specification
- formal modelling in FOOM is undertaken in the specification language Object-Z. As with the other types of modelling techniques, the general guideline is to model formally where it seems appropriate. In the case of formal modelling, this is most often where conflict is known or suspected to exist, either between or within problem statements in particular. The power of formal modelling lies in the depth of understanding generated, and its precision in identifying what may be subtle conflict between world views. Not all requirements (in particular, "useability") can be represented formally of course. The expected path is to progress from the development of fragments of Object Z to the gradual construction of a complete formal model of the requirements for a relevant human activity system. We do not use OCL (Warmer and Kleppe, 1999) as it is not a

complete formal modelling language but rather a mechanism for expressing additional (precise) statements regarding classes in a system.

The expected outcome of the information analysis and modelling activities is the identification of ambiguities and imprecisions which, with the assistance of the information analyst and the actors within the problem context, will be resolved.

debate and validation In this activity (as in information analysis), the primary participants are the actors within the problem context. The role of the information analyst is to help identify (but not normally to solve) conflict both between and within problem statements generated (formal or otherwise) from the modelling activity. The formal specifier's role within this process is to assist in evaluating the implications of formal problem statements, and to facilitate decision making by the actors within the problem context (the owners of the problem)

The techniques and notations associated with the conduct of these activities may be grouped by the areas described previously: socio-organisational theory, object orientation, and formal specification techniques.

In summary then, the techniques included within FOOM are:

- the formulation of root definitions to represent relevant human activity systems, from the Soft Systems Methodology. These may or may not be expressed with rich pictures
- the static modelling tools associated with MOSES or OPEN: association, aggregation, inheritance, etc
- dynamic modelling tools designed especially for FOOM: the event chain notation and object collaboration diagrams
- the Object-Z specification language
- textual and ad hoc diagrammatic modelling.

SUMMARY

The use of formal specification techniques in the information systems domain offers the potential of improved relevance of software developed by offering:

- the provision of a deeper insight into the system being specified - a formal specification prevents the creation of inherent contradictions caused by redefining the “meaning” of portions of the specification without corresponding changes to the specification itself
- a mechanism allowing study and analysis of the specification, thus offering reliable predictions of behaviour; and leading to a basis for system acceptance testing.

This paper has described a development methodology, OPEN/FOOM, which incorporates formality into the requirements engineering process for commercial information systems development. Specifically, it described the issues associated with incorporating formal techniques into the requirements engineering process, and describes the benefits of supporting this approach with the third generation object-oriented methodology known as OPEN.

REFERENCES

Boehm B.W. (1976) Software engineering, *IEEE Transactions on Computers* C-25(12): 1226-1241.

Booch G. (1994) *Object-Oriented Analysis and Design with Applications*, 2nd ed, Benjamin Cummings, Redwood City, California.

Bowen J. and Hinchey M. (1995) 10 commandments of formal methods, *IEEE Computer* 28(4):56-63.

Bowen J. and Stavridou V. (1993) The industrial take-up of formal methods in safety-critical and other areas *in* Woodcock and Larsen (eds), *FME'93: Industrial-Strength Formal Methods*, Springer-Verlag, pp183-195.

Brooks F. (1994) No silver bullet: Essence and accidents of software engineering, *Computer*, 10-18.

Burrell G. and Morgan G. (1979) *Sociological paradigms and organisational analysis*, Heineman, London.

Checkland P.B. (1995) Soft systems methodology and its relevance to the development of information systems, *in* Stowell (1995).

Checkland P.B. (1981) *Systems Thinking, Systems Practice*, Wiley, Chichester.

Checkland P.B. and Scholes J. (1990) *Soft Systems Methodology in Practice*, Wiley.

Craigen D., Gerhart S. L. and Ralston T.J. (1993) *An international survey of industrial applications of formal methods*, Tech Rpt NIST GCR 93/626- V1 & 2, Atomic Energy Control Board of Canada, US National Institute of Standards and Technology, and US Naval Research Laboratories.

Cunningham R.J., Finkelstein A., Goldsack S., Maibaum T. and Potts C. (1985) Formal requirements specification - the Forest project, *Proceedings of the International Workshop on Software Specification and Design*, London, pp. 186-191.

Duke R., King P. and Smith G. (1991) Combining formal methods with object-oriented design: An emerging trend in software engineering, *Proc Australian Computer Conference-ACC'91*.

Firesmith D.G. (1995) Use cases: the pros and cons, Report on Object Analysis and Design, 2(2),2-6.

Fowler D.C., Swatman P.A. and Wafula E. (1995) Formal methods in the IS domain: introducing a notation for presenting Object-Z specifications, *Object Oriented Systems*, 2(2).

Fowler D.C. (1996) Formal Methods in a Commercial Information Systems Setting: The FOOM Method, PhD thesis, Centre for IS Research, Swinburne University, Melbourne, Victoria.

Fraser M., Kumar K. and Vaishnavi V. (1994) Strategies for incorporating formal specifications in software development, *Communications of the ACM* 37(10): 74-86.

Gerhart S., Graigen D. and Ralston T. (1994) Experience with formal methods in critical systems, IEEE Software 11(1): 21-28.

Gibbs W. (1994) Software's chronic crisis, Scientific American pp. 72-81.

Hall J.A. (1990) Seven myths of formal methods, *IEEE Software* 7(5): 11-19.

Henderson-Sellers B. (1997) *A BOOK of Object Oriented Knowledge*, 2nd ed, Prentice Hall, NJ.

Henderson-Sellers B. and Edwards J. M. (1994) *BOOKTWO of Object-Oriented Knowledge: The Working Object*, Prentice Hall, Sydney.

Henderson-Sellers B., Simons A. and Younessi H. (1998) *The OPEN Toolbox of Techniques*, Addision-Wesley, New York.

Hinchey M. and Bowen J. (eds) (1995) *Applications of Formal Methods*, International Series in Computer Science, Prentice Hall.

Jacobson I., Ericsson M. and Jacobson A. (1995) *The Object Advantage: Business Process Reengineering with Object Technology*, Addison-Wesley, Wokingham, England.

Korson T. (1998) The misuse of use cases (managing requirements), *Object Magazine*, 8(3):18-20.

Lano K. and Haughton H. (1994) *Object-Oriented Specification Case Studies*, Prentice Hall, Englewood Cliffs.

Lin M. and Henderson-Sellers B. (1998) Adapting the open methodology for web development, *Proceedings of the BCS Information Systems Methodolgy Specialist Group Conference*, University of Manchester, June (Springer Verlag, in press).

Mander K. C. and Polack F. (1995) Rigorous specification using structured systems analysis and Z, Information and Software Technology 37(5-6): 285-291.

Orlikowski W. J. and Baroudi J. (1991) Studying information technology in organisations: research approaches and assumptions., *Information Systems Research* 2(1): 1-28.

Polack F. and Mander K. C. (1994) Software quality assurance using the SAZ method, *in* J. P. Bowen and J. A. Hall (eds), *Z User Workshop*, Cambridge, Workshops in Computing, Springer-Verlag, pp. 230-249.

Rumbaugh J. et al. (1991). *Object-Oriented Modelling and Design*, Prentice Hall, Englewood Cliffs.

Shlaer S. and Mellor S. (1992) *Object Lifecycles: Modelling the World in States*, Prentice Hall, Englewood Cliffs, N.J.

Sawyer P., Sommerville I. And Viller S. (1999) Capturing the benefits of requirements engineering. *IEEE Software*, March/April, 78-85.

Spivey J.M. (1992) *The Z Notation: A Reference Manual*, 2nd ed, Prentice Hall, Hemel Hempsted, England.

Stowell F.A. (ed) (1995) *Information Systems Provision: the Contribution of Soft Systems Methodology*, McGraw-Hill Book Company Europe, Berkshire, England.

Swatman P.A. (1996) *Formal object-oriented method: Foom*, in Kilov H.and Harvey W. (eds), Specification of Behavioural Semantics in Object-Oriented Information Systems, Kluwer.

Swatman P. A. (1992) Increasing Formality in the Specification of High-Quality Information Systems in a Commercial Context, PhD thesis, Curtin University of Technology, School of Computing, Perth, Western Australia.

Swatman P.A., Fowler D.C. and Gan C.Y.M. (1992) Extending the useful application domain for formal methods, in J. E. Nicholls (ed.), *Z User Workshop*: York 1991, Workshops in Computing, Springer Verlag, London.

Swatman P.A. and Swatman P.M.C. (1992) Formal specification: An analytic tool for (management) information systems, *Journal of Information Systems* 2(2): 121-160.

Vessey I. and Glass R. (1998) Strong vs weak approaches to systems development, *Communications of the ACM* 41(4): 99-102.

Wafula E.N. and Swatman P.A. (1996) FOOM: a diagrammatic illustration of Object-Z specifications, *Object Oriented Systems*, 3(4), 215-242.

Winograd T. and Flores F. (1986) *Understanding Computers and Cognition: A New Foundation for Design*, Ablex, Norwood, N.J.

Wirfs-Brock R., Wilkerson B. and Wiener L. (1990) *Designing Object-Oriented Software*, Prentice Hall, Englewood Cliffs, N.J.

COPYRIGHT