Rethinking Systems Thinking

Shayne Flint
Department of Computer Science
The Australian National University, Canberra, ACT 0200, Australia
shayne.flint@anu.edu.au

Abstract

Systems Thinking refers to a set of approaches that can be used to learn about and make decisions regarding improvements to dynamically complex systems. They are distinguished from other approaches by their focus on the whole and the study of interactions among the parts of a system, rather than the parts themselves. While a focus on interactions helps in understanding complex systems and identifying appropriate improvements, it is necessary to use detailed knowledge of the parts and other aspects of a system to implement any improvements. This paper addresses this issue by introducing a novel Systems Thinking approach which uses detailed knowledge of the parts to both understand the whole, and to build the systems required to implement necessary improvements.

Keywords

Systems thinking, aspect-oriented thinking, knowledge management, software-intensive systems, systems engineering, model-driven engineering, model-driven development

INTRODUCTION

A great deal of human activity is dedicated to '*solving problems*'. However, problems usually emerge within dynamically complex (Casti 1979) environments of co-evolving technology, people, processes and other problems (Ackoff 1999). They are often affected by the progress of other problems and changes throughout the broader context. Because of this, individual problems are often difficult to isolate and independently solve. Attempts to do so often lead to inappropriate development, use and destruction of systems.

It is generally accepted that dynamically complex sets of interacting problems, or *Problem Situations* (Checkland 1981), are best dealt with by continuously *improving* the whole rather than by attempting to *solve* individual problems (Ackoff 1999; Checkland 1981). In order to make appropriate improvements, it is first necessary to fully understand all aspects of a problem situation and the likely impact of any proposed improvements.

Systems Thinking (Churchman 1968; Kramer 1977; Checkland 1981; Senge 1992) refers to a set of approaches that can be used to learn about and make decisions regarding improvements within dynamically complex systems such as *Problem Situations*. They are distinguished from other approaches by their focus on the whole and the study of interactions among the parts of a system, rather than the parts themselves. It is by studying these interactions that emergent properties of an entire system, including the likely impact of a change on the whole, can be understood.

While existing *Systems Thinking* approaches can help people understand *Problem Situations* and identify appropriate improvements, they do not generally address the implementation of such improvements. To do this, it is necessary to build, operate and retire systems using deep and detailed knowledge about a broad range of subject matters within a reductionist framework.

The purpose of this paper is to contribute to development of systems thinking approaches that not only deal with understanding complex problem situations, but also the development of systems needed to implement necessary improvements. This is acheived by providing an overview of Aspect-Oriented Thinking (AOT), a novel *Systems Thinking* approach to managing and using multi-disciplinary reductionist knowledge to build and operate systems required to learn about complex problem situations, identify appropriate improvements and to implement and evaluate such improvements.

ASPECT-ORIENTED THINKING

Aspect-Oriented Thinking was developed within an engineering context to improve the effectiveness of large complex technical systems. Systems of interest included engineering design, scientific computing, and national security. The focus was on improving productivity and ensuring that systems were developed and operated with a full understanding of their impact over time and space. This necessitated the development of a multi-disciplinary systems thinking approach to engineering.

As the development of AOT progressed, it became clear that the approach could form the basis of a generalised systems thinking approach to improving a broad range of problem situations involving the study, development and use of technical, social, cutural, legal, political, economic, natural and other systems.

The remainder of this paper describes AOT in terms of the concepts involved and the process used. A more detailed description can be found in Flint (2008). Earlier work, including a software development case study, can be found in Flint (2006).

Conceptual Model

The four main concepts involved in AOT are depicted in Figure 1 and can be described as follows:

- **Domains and Domain Models.** In order to help people fully understand a *Problem Situation*, AOT applies the principle of separating concerns (Dijkstra 1982) in multiple dimensions. The various subject matters (*Domains*) involved in a *Problem Situation* are considered separately. For example, within the context of emergency services, *Domains* such as policing, fire-fighting and medical care, along with cross-cutting *Domains* such as transportation, communications, and financing would be considered separately. Knowledge about each *Domain* is captured in one or more autonomous *Domain Models* that each represents a particular view of a given *Domain* within a *Problem Situation*.

  Because they are autonomous, most *Domain Models* will be independently developed and maintained by engineers, scientists, sociologists, psychologists, lawyers, philosophers, economists and others, using languages and techniques with which they are familiar. They will often be developed within a reductionist framework and will exist in various forms including publications, software, data and other systems.

- **Aspect-Oriented Specification Archetypes.** *Aspect-Oriented Specification Archetypes* represent a key concept in AOT. They are used to model the archetypical ways in which domains interact or can interact within a given type of *Problem Situation*. That is, they capture the results of *Systems Thinking*.

  For example, an *Aspect-Oriented Specification Archetype* dealing with aspects of ecology might describe the archetypical ways in which plants of a particular type respond to fire. These descriptions take the form of a set of prototypes, patterns or templates for describing specific interactions in *Aspect-Oriented Specifications* described below.

- **Aspect-Oriented Specifications.** *Aspect-Oriented Specifications* describe real, hypothetical or abstract systems including those required to learn about and improve a *Problem Situation*. They identify the domains involved (such as sub-systems, theories, models and laws), and the ways in which they interact.

  Aspect-Oriented Specifications are formed in accordance with *Aspect-Oriented Specification Archetypes*. That is, *Aspect-Oriented Specification Archetypes* describe the archetypical ways in which domains interact to form systems, while *Aspect-Oriented Specifications* describe specific systems in terms of those archetypes.

- **Implementation processes.** *Implementation Processes* interpret *Aspect-Oriented Specifications* and generate systems of various kinds including software, hardware, plans, procedures, simulations, models and documentation. Because of the relationship between *Specifications*

and *Specification Archetypes* described above, it is possible to develop *Implementation Processes* which can process any *Specification* that complies with a given *Specification Archetype*.

While *Implementation Processes* can be executed manually, automation can result in significant productivity gains over conventional approaches to system development.

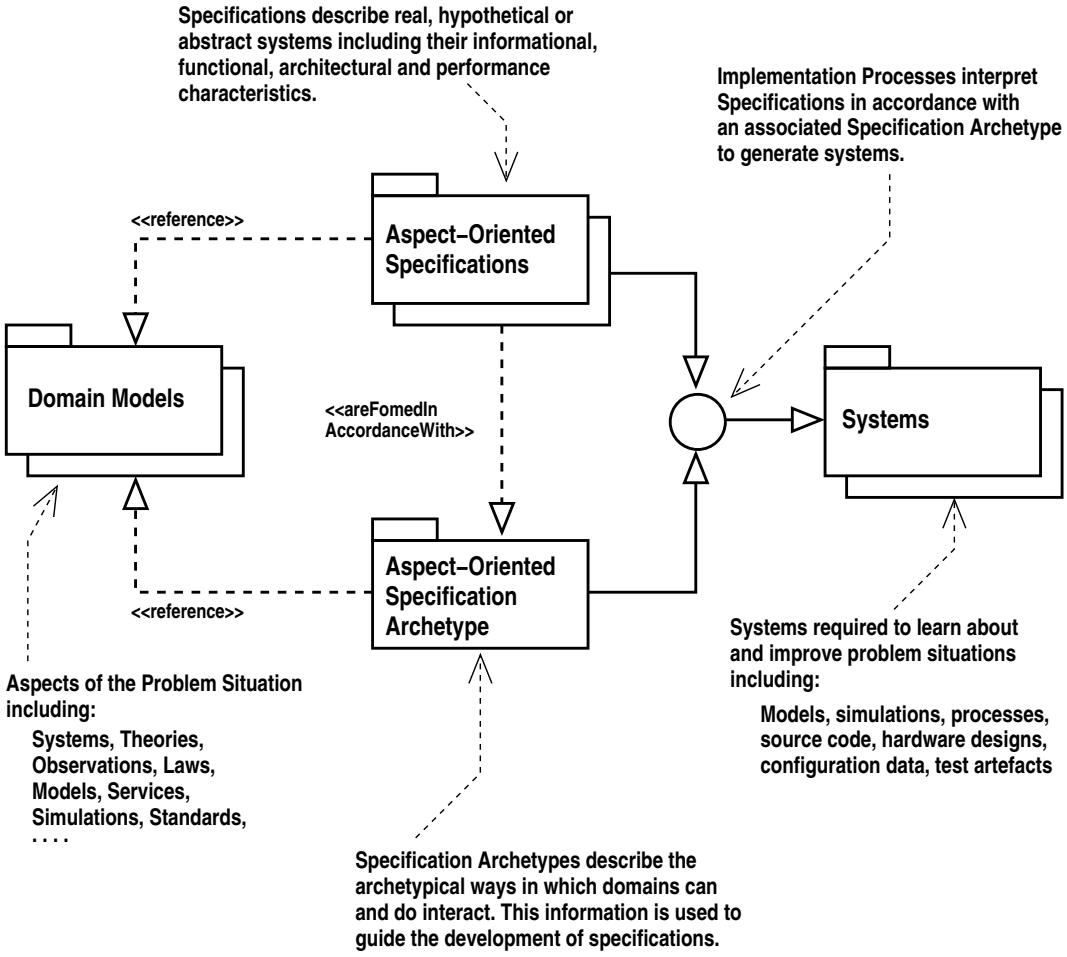

Figure 1. *Aspect-Oriented Thinking concepts.*

Process Model

AOT is a *continuous process of learning and improvement* based on Boyd's *Observe-Orient-Decide-Act (OODA)* decision loop (Boyd 1986) as depicted in Figure 2. Within the context of improving a *Problem Situation*, information is collected and recorded during the *Observation* phase. During *Orientation*, this information is used to develop an understanding of a *Problem Situation* and to identify necessary improvements. Options regarding the development, operation, modification and retirement of systems required to *implement* these improvements are explored during the *Decision* phase. Selected options are then implemented during the *Act* phase. The loop repeats (now including the actual impact of earlier changes, as well as changes independantly made to other parts of the problem situation) until all stakeholders agree that no further improvements are necessary.

Note that the OODA loop described here will be operating at many places and at many levels within a single *Problem Situation*. Some processes will involve the use of AOT and some will not. They will be operating at different rates, under different sets of constraints and will be changing the *Problem Situation* in ways that impact the operation of other OODA loops.

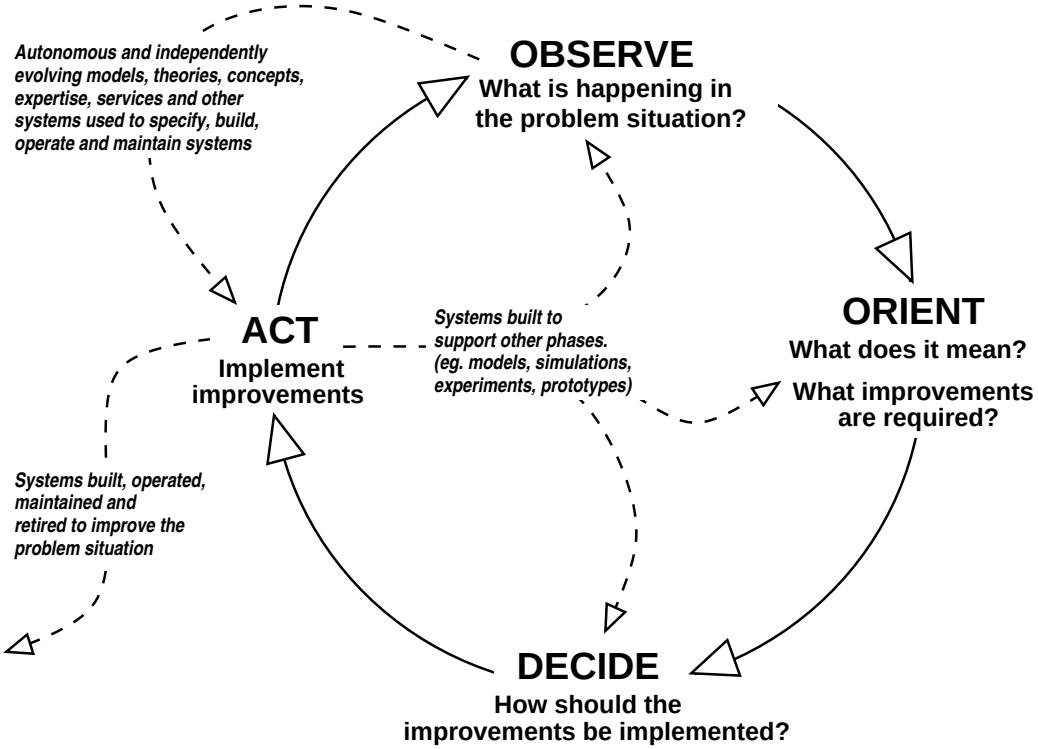

Figure 2. *The process of AOT is based on Boyd's Observe-Orient-Decide-Act (OODA) loop. It aims to understand complex problems situations, identify necessary improvements and then implement them by building and operating systems of various kinds.*

Each phase of the AOT process is summarised below:

- **Observation.** During *Observation*, the various subject matters involved in a *Problem Situation* are identified as *Domains*. Autonomous *Domain Models* are then identified (reused) or developed to represent a set of applicable views of each *Domain*.

  Specifications and *Specification Archetypes* can also be formed during *Observation* to capture and communicate common patterns of organisation, interaction, behaviour and other aspects evident throughout a *Problem Situation*, and how these patterns manifest themselves in systems that already exist within the *Problem Situation*.

- **Orientation.** The aim of *Orientation* is to make sense of a *Problem Situation* and to identify possible improvements. Understanding a *Problem Situation* is done without concern for the construction of any particular system and will involve understanding each *Domain* as well as the ways in which they are, and can be, woven together within a *Problem Situation*.

  To understand each *Domain*, various traditional model analysis techniques can be used. For example, simulations and executable modeling languages such *System Dynamics* (Forrester 1961) may be used to understand the dynamics of a particular subject matter.

  In order to understand interactions between different *Domains*, *Specification Archetypes* and corresponding *Specifications* developed during the *Observation* phase can be analysed statically and/or dynamically. Static analysis involves traditional activities such as ensuring that the *Specifications*, *Specification Archetypes* and referenced *Domain Models* are properly formed and that they reflect appropriate levels of agreement between all the stakeholders involved. Dynamic analysis involves the use of simulations. If appropriate domain modeling languages such as *System Dynamics* (Forrester 1961) or the *Business Process Modeling Notation* (BPMN) (Object Management Group 2006) are used, *Specifications* can be implemented as simulations which can be used to explore the dynamics of systems involved in a *Problem Situation*.

Once the *Domains* and *Systems* involved in a *Problem Situation* are fully understood, stakeholders can work towards agreement regarding the identification of necessary improvements.

- **Decision.** During the *Decision* phase, options for implementing improvements are identified and evaluated. *Domain Models* developed during *Observation* and *Orientation* can be reused to form *Specification Archetypes* and *Specifications* for systems required to evaluate each option. Initially, these *Specifications* may relate to simulations and other analysis tools which can be used to explore the likely impact of each option on the *Problem Situation*.
- **Action.** The purpose of the *Action* phase is to implement decisions made during the *Decision* phase. This is achieved by extending *Specifications* for selected options to include architecture, design, implementation, testing and deployment concerns. Appropriate *Implementation Processes* are then developed (or reused) and executed to translate *Specifications* into operational systems and other artefacts.

Advantages and Limitations

Advantages of the AOT approach to *Systems Thinking* include:

- **Implementing Improvements.** While *Systems Thinking* approaches support the identification of appropriate improvements within complex systems, they provide little support for developing and operating the systems required to implement such improvements. AOT represents an integrated *Systems Thinking* approach which can be used to both identify and implement improvements within complex systems.
- **Multi-disciplinarity.** The separation of *Domain Models* from system specification supports multi-disciplinarity without the need for common languages, techniques and tools.

  Domain Models can be developed in any appropriate language using any appropriate technique. AOT does not impose any constraints on the way knowledge is developed or represented. This allows each discipline to independently build their own bodies of knowledge, languages, techniques and expertise within a reductionist framework (as they, in fact, do). The concepts of *Specification* and *Specification Archetype* allow us to weave together this independently developing knowledge and expertise without weakening the depth of knowledge involved or looking for some lowest common denominator. That is, AOT facilitates a multi-disciplinary approach that protects the depth of knowledge developed by each discipline.
- **Reuse.** AOT supports extensive reuse. *Domain Models*, *Specifications* and *Specification Archetypes* will evolve over time and are extensively reused throughout the AOT process. Initially, *Domain Models* will be used to develop a static understanding of a *Problem Situation*. These *Domain Models* will be reused to form *Specifications* and *Specification Archetypes* to describe more complex aspects of a *Problem Situation*. These *Specifications* and *Specification Archetypes* might then be reused and extended to deal with simulation. These simulations would then be used to learn more about the dynamics of a *Problem Situation* and to identify any necessary improvements. The same *Specifications* and *Specification Archetypes* may then be modified again to reflect various options for implementing these improvements. Finally, they would be modified to form *Specifications* for operational systems which will be deployed to improve a *Problem Situation*.
- **Stakeholder Agreement.** AOT is tolerant of stakeholder disagreement.

  Within a universal context, or even within the context of a particular *Problem Situation*, reaching consensus among people from a wide range of disciplines and perspectives regarding the meaning, completeness, correctness and content of *Domain Models*, is very unlikely. Fortunately, AOT does not require agreement within such broad contexts. Instead, the need for agreement lies at the level of forming *Specification Archetypes* and corresponding *Specifications*. The context at this level is narrow and relates to a specific purpose. It involves fewer stakeholders who only need to reach agreement regarding the meaning of domain knowledge for that purpose.

- **Controlled Agility.** A key concept underpinning the OODA loop is that decision makers who cycle through their OODA loop more rapidly than others, will have a competitive advantage. Slower decision makers will make decisions based on observations and orientation that may be invalid by the time such decisions are implemented. As a result, their actions are often inappropriate and costly. More importantly, because the decisions of others may be more agile, relentless and based on relevant and up-to-date observations and orientation, the world may appear chaotic to slower decision makers.

  The automation of *Implementation Processes* along with the AOT approach to capturing and reusing knowledge in *Domain Models*, *Specification Archetypes* and *Specifications*, increases the agility with which the AOT process (OODA loop) can be executed. This can, in turn, improve the correctness of decisions, as well as the responsiveness of an organisation to external change and disturbances.

Current limitations of the AOT approach and strategies to deal with them, include:

- **Evaluation.** AOT is a novel approach to *Systems Thinking* that has only been used on small case studies. There is clearly a need to explore the effectiveness of AOT on a larger scale. To do this, we have developed a strategy to run AOT projects within the context of larger projects. One such project is under way within the environmental sciences domain.
- **Tool Support.** There is no effective tool support for AOT. A prototype tool has been developed to support an earlier version of the approach, but it has proved cumbersome and difficult to use, especially for non-technical users. More recently, some thought has gone into the development of a new generation web-based tool, but actual development has not yet commenced.

CONCLUSION

Traditional *Systems Thinking* approaches can be used to understand complex *Problem Situations* and to identify appropriate improvements. Such approaches do not, however, provide effective support for implementing such improvements. This paper has presented a overview of research underlay to develop *Aspect-Oriented Thinking*, a novel *Systems Thinking* methodology that incorporates the use of detailed knowledge, developed within a reductionist framework, to build, operate, modify and retire the systems necessary to learn about and improve dynamically complex *Problem Situations*.

It is hoped that this overview will contribute to an ongoing discussion about the development, operation and retirement of systems within the context of *Systems Thinking*.

REFERENCES

Ackoff, R. L. (1999). *Ackoff's Best, His Classic Writings on Management*. New York: John Wiley & Sons.

Boyd, J. R. (1986). Patterns of conflict (unpublished). viewed 19 June 2008, <http://www.d-n-i.net/boyd/pdf/poc.pdf>.

Casti, J. (1979). *Connectivity, complexity, and catastrophe*. Chichester, England: J. Wiley.

Checkland, P. (1981). *Systems Thinking, Systems Practice*. New York: J. Wiley.

Churchman, C. W. (1968). *The Systems Approach*. New York: Dell Publishing Inc.

Dijkstra, E. W. (1982). EWD 447: On the role of scientific thought. In *Selected writings on Computing: A Personal Perspective*, pp. 60–66. Springer-Verlag.

Flint, S. R. (2006, July). *Aspect-Oriented Thinking: An approach to bridging the disciplinary divides*. Ph. D. thesis, Australian National University, Canberra, Australia. Available from *Australian Digital Thesis Program*, http://thesis.anu.edu.au/public/adt-ANU20080731.204756/index.html.

Flint, S. R. (2008, September). A Model-Driven Approach to Systems-of-Systems Engineering. In *Proceedings of the 2008 Systems Engineering Test and Evaluation conference (SETE 2008)*, Canberra, Australia.

Forrester, J. W. (1961). *Industrial Dynamics*. Cambridge, MA: The MIT Press.

Kramer, N. J. T. A. (1977). *Systems thinking: concepts and notions*. Leiden: Martinus Nijhoff.

Object Management Group (2006). Business Process Modeling Notation Specification, version 1.0. viewed 19 June 2008, <http://www.omg.org>.

Senge, P. M. (1992). *The Fifth Discipline*. Milsons Point NSW: Random House Australia.

COPYRIGHT

Shayne Flint ©2008. The author/s assign Edith Cowan University a non-exclusive license to use this document for personal use provided that the article is used in full and this copyright statement is reproduced. Such documents may be published on the World Wide Web, CD-ROM, in printed form, and on mirror sites on the World Wide Web. The authors also grant a non-exclusive license to ECU to publish this document in full in the Conference Proceedings. Any other usage is prohibited without the express permission of the authors.