Agent-Based Simulation: Social Simulation and Beyond

Bruce Edmonds (bruce@cfpm.org)
Emma Norling (norling@acm.org)
Centre for Policy Modelling
Manchester Metropolitan University
Aytoun Building, Aytoun St, Manchester, M1 3GH, United Kingdom

Abstract

Our interest in agent-based simulation is for *social simulation*, where the society-level outcomes emerge from the interaction of individuals. In this tutorial, we aim to introduce the core concepts of agent-based social simulation, illustrated by a range of examples, before walking through a specific example with the participants so that they can experience these issues at first hand.

Keywords: social simulation, social theory, agent-based modelling

What is Agent-Based Simulation?

Agent-based simulation (ABS) represents each individual with a separate encapsulated object in a simulation. Beyond this, the definition of an "agent" varies quite widely, but in general they are seen to be autonomous, pro-active entities. Simulation outcomes emerge from the interactions between these entities, and often even quite simple interactions can give rise to complex system dynamics.

The individuals that agents represent in a simulation need not be humans, and could be social actors of any type. Examples of entities that have been represented by agents in simulations range from individual cells and bacteria through to multi-national corporations. Typically though in social simulation we are interested in modelling each individual person with a single agent. At the same time, we are often interested in modelling the interactions of large numbers of individuals, and this forces a trade off between the detail of the individual models and the number of entities that can be modelled.

Thus, while it is desirable for the agents to include models of various aspects of cognition (such as decision making, learning, belief representation, autonomous goals), it is necessary to pare them down to the bare minimum required to model the social interactions of interest. By the standards of cognitive models many of the programs internal to each agent might be fairly simple, although some researchers in this area are investigating ways of including more detailed models of individuals within this type of simulation.

Simulating Societies

Our interest in ABS is to simulate how humans (or other social entities) might interact: for example, how complex coordination might be achieved through the interaction of essentially selfish agents (Edmonds, 2006). Some of these models can be very detailed, including many different aspects of a particular observed social situation. In this case the result is more like a dynamic description within a simulation – a distributed representation that may incorporate many different kinds of evidence.

Emergent Behaviour

At the same time, complexity science has repeatedly shown how the interaction of fairly simple agents can result in complex ("emergent") outcomes. Thus, one stream of research in ABS is looking at how social systems might be understood in this way. These tend to be quite abstract simulations with very simple agents, which are intended to encapsulate a general social theory, rather than to represent any particular observed social phenomena.

Applying Social "Rules" to Other Networks

One outcome of the study of emergent behaviour in human societies has been to transfer these principles to other social systems. For example, when systems of independently programmed computers interact in a network, many of the same issues (trust, reputation, coordination etc.) that occur in human societies are found to be important. The previously mentioned work on cooperation between self-interested individuals, for example, has been used to develop algorithms for peer-to-peer computing systems that are robust against "cheaters" (Hales, 2006).

Outline of the Tutorial

This tutorial introduces the main ideas of ABS, highlighting the difficulties as well as the strength of these issues, drawing on many examples of ABS, from complex specific simulations, up to highly abstract simulations that encapsulate social theories. In the second half of the tutorial, these ideas will be illustrated through the use of a concrete example. Depending on the existing skills of the participants, there will be opportunities to implement their own realisation of this example.

Acknowledgements

Some material in this tutorial is based upon a tutorial presented by Franziska Klügl and Emma Norling at the European Agent Systems Summer School, 2006.

References

Edmonds, B (2006). The Emergence of Symbiotic Groups Resulting from Skill-Differentiation and Tags. *Journal of Artificial Societies and Social Simulation* 9(1)10 <http://jasss.soc.surrey.ac.uk/9/1/10.html>.

Hales, D. (2006) Emergent Group-Level Selection in a Peer-to-Peer Network. *Complexus* 2006;3:108-118

Cognitive Modelling with the Neural Engineering Framework

Chris Eliasmith (celiasmith@uwaterloo.ca)
Terrence C. Stewart (tcstewar@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
Waterloo, Ontario, Canada, N2L 3G1

Abstract

The Neural Engineering Framework (NEF; Eliasmith and Anderson, 2003) provides a general methodology for developing efficient and realistic neural models that perform a specified task. The framework consists of three quantified principles, one for each of representation, computation, and dynamics in neural systems. Adopting these principles provides a method for generating connection weights between groups of neurons that represent and transform state variables. In short, the NEF provides a neural compiler: a method for taking a high-level description of a neural system and deriving a plausible organization of realistic neurons that realize this system. Our tutorial introduces the principles of the NEF and demonstrates how they apply to cognitive modeling. This is done through the use of Nengo, a GUI neural simulation system, which supports an adjustable level of neural accuracy, Python scripting, and the analysis of the resulting models.

Keywords: Nengo; neural engineering; neural representation; control theory; neural cognitive modelling

The Neural Engineering Framework

As cognitive models become more complex, there is an increased demand for details at both the low and high levels. Traditionally, focus in cognitive modeling has been on higher levels of abstraction. As a result, researchers typically posit a high-level organizational structure which allows them to consider the information that needs to be represented and the transformations that are required for implementing hypothesized algorithms. Ideally, however, a cognitive model should also make detailed predictions as to the firing rates of neurons implementing the model, their tuning curves, connectivity, neurotransmitters, and other properties.

The Neural Engineering Framework (or NEF; Eliasmith and Anderson, 2003) provides a novel approach to addressing this typical gap in cognitive modeling. It is based on three principles of neural engineering:

1. Neural representations are defined by the combination of nonlinear encoding (exemplified by neuron tuning curves) and weighted linear decoding.
2. Transformations of neural representations are functions of variables that are represented by neural populations. Transformations are determined using an alternately weighted linear decoding.
3. Neural dynamics are characterized by considering neural representations as control theoretic state variables. Thus, the dynamics of neurobiological systems can be analyzed using control theory.

Each of these principles is considered under the assumption that neural systems are subject to significant amounts of noise. Therefore, any analysis of such systems must account for the effects of noise.

The core idea of the NEF is to consider any cognitive system as containing a large number of representations which change over time. How these representations change is dependent both on the external stimuli and on the other representations within the system. A particular neural group can represent (via its spike pattern) a single scalar, a vector, or even a function. These representations are inherently noisy, and accuracy will be dependent on various neural properties (although representational error has been shown to be inversely linearly related to the number of neurons used).

To understand how these representations change (i.e. define a transformation of a representation), the NEF provides methods for defining weighted axonal projections. For instance, a given group might represent the product of the values being represented by two other groups which are projected to it (i.e. $x(t) = y(t)*z(t)$, where each variable is represented by a neural population). Importantly, we can use the NEF to derive the linearly optimal connection weights to perform a wide variety of linear and nonlinear transformations. Doing so makes it clear that the accuracy of these transformations is intimately related to the observable tuning curves of the neurons involved. This leads to models that are orders of magnitude more efficient than other approaches to neural representation, and which are a closer match to observed neurological data (e.g. Conklin & Eliasmith, 2005; Fischer, 2005).

Applications

Initially, the main applications of this approach were in the domains of sensory and motor systems. This has included the barn owl auditory system (Fischer, 2005), rodent navigation (Conklin & Eliasmith, 2005), escape and swimming control in zebrafish (Kuo & Eliasmith, 2005), and the translational vestibular ocular reflex in monkeys (Eliasmith et al., 2002). However, these same principles are now being applied to higher-level cognitive models. A direct extension of the visual working memory model (Singh & Eliasmith, 2006) has led to a neural model of the ACT-R goal buffer (Stewart, Tripp, & Eliasmith, 2008). More crucially, the use of Vector Symbolic Architectures (Gayler, 2003) has allowed for the representation and manipulation of structured symbol trees by these neural models. This neurally realistic cognitive architecture (Stewart & Eliasmith, 2009a) resulted in a model of the Wason card task (Eliasmith, 2005) and ongoing work producing an efficient production system using realistic neural constraints (Stewart & Eliasmith, 2008; 2009b).

The NEF provides an exciting new tool for cognitive modelers as it provides a technique for producing direct neural predictions from a given high-level algorithmic description of a cognitive model. Furthermore, it leads to important theoretical results as to the relationships between neural properties and the high-level algorithms they are capable of implementing (e.g. the relationship between neurotransmitter re-uptake rate and the time constant of neural transformations).

These consequences are also very general, as the NEF provides techniques that can be applied to any cognitive model. It provides a structure for organizing the high-level description of a model, such that it can be implemented by realistic spiking neurons, providing meaningful data in terms of the expected spike patterns, time course, and accuracy. We have made use of it in a wide variety of contexts, and we have developed tools that support the creation and analysis of these models. These tools can be applied to many existing models to incorporate low-level neural details into existing modeling research.

Software and Simulations

We have developed Nengo <nengo.ca>, a freely available open-souce Java-based neural simulator that supports the NEF. This allows for hand-on examples of the theoretical concepts underlying the NEF. Using a point-and-click interface, we can create neural group, configure them to represent scalars and vectors, adjust their neural properties, and simulate their spiking activity over time. We can also connect these neural groups via synapses so as to perform linear and nonlinear transformations on these values, and store information over time. These are the basic mechanisms required for a wide range of algorithms, and form the basis for our models of sensorimotor systems and working memory. Nengo can also be programmed using a Python interface, allowing for quick creation of complex models (Stewart, Tripp, & Eliasmith, 2009).

Furthermore, these basic tools can be used to implement the theory of Vector Symbolic Architectures (Gayler, 2003) using NEF. This involves using high-dimensional fixed-length vectors to represent symbols and symbol trees. The nonlinear operation of circular convolution is used to manipulate these symbol trees. This can be seen as a non-classical symbol system, capable of performing the operations required for symbolic cognition. The result is a scalable and efficient neural cognitive architecture, constructed from these basic neural components.

References

Conklin, J. & Eliasmith, C. (2005). An attractor network model of path integration in the rat. *Journal of Computational Neuroscience 18*, 183-203.

Eliasmith, C. (2005). *Cognition with neurons: A large-scale, biologically realistic model of the Wason task.* 27th Annual Meeting of the Cognitive Science Society.

Eliasmith, C. & Anderson, C. (2003). *Neural Engineering: Computation, representation, and dynamics in neurobiological systems.* Cambridge: MIT Press.

Eliasmith, C., Westover, M.B., & Anderson, C.H. (2002). A general framework for neurobiological modeling: An application to the vestibular system. *Neurocomputing 46*, 1071-1076.

Fischer, B. (2005). *A model of the computations leading to a representation of auditory space in the midbrain of the barn owl*. PhD thesis. Washington University in St. Louis.

Gayler, R. (2003). *Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience.* ICCS/ASCS International Conference on Cognitive Science, 133-138.

Kuo, D. & Eliasmith, C. (2005). Integrating behavioral and neural data in a model of zebrafish network interaction. *Biological Cybernetics*. *93*(3), 178-187.

Singh, R., & Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *Journal of Neuroscience*, *26*, 3667-3678.

Stewart, T.C. & Eliasmith, C. (2008). *Building production systems with realistic spiking neurons.* 30th Annual Meeting of the Cognitive Science Society.

Stewart, T. C. & Eliasmith, C. (2009a). *Compositionality and biologically plausible models*. In Oxford Handbook of Compositionality, W. Hinzen, E. Machery, M. Werning, eds. (Oxford University Press).

Stewart, T. C., & Eliasmith, C. (2009b). *Spiking neurons and central executive control: The origin of the 50-millisecond cognitive cycle*. 9th International Conference on Cognitive Modelling.

Stewart, T. C., Tripp, B., & Eliasmith, C. (2008). *Supplementing Neural Modelling with ACT-R*. 15th Annual ACT-R Workshop.

Stewart, T. C., B. Tripp, & Eliasmith, C. (2009). Python scripting in the Nengo simulator. *Frontiers in Neuroinformatics. 3*(7), 1-9.

EPAM/CHREST Tutorial: Fifty Years of Simulating Human Learning

Peter C. R. Lane (peter.lane@bcs.org.uk)
School of Computer Science, University of Hertfordshire, College Lane, Hatfield AL10 9AB, UK

Fernand Gobet (fernand.gobet@brunel.ac.uk)
School of Social Sciences, Brunel University, Cleveland Road, Uxbridge, Middlesex, UB8 3PH, UK

Keywords: CHREST; EPAM; cognitive modelling; chunking

Overview

This tutorial covers a tradition of symbolic computational modelling known as EPAM/CHREST, with its first member, EPAM (Elementary Perceiver and Memoriser) developed by Edward Feigenbaum in 1959. EPAM was used to construct models of a variety of phenomena, providing the impetus to develop the chunking theory (Chase & Simon, 1973; Gobet et al., 2001), which has been an important component of theories of human cognition ever since.

The history of computational modelling includes a variety of approaches to describe human behaviour. The benefits of encoding a theory as a computational model include a precise definition of how the behaviour is to be explained, and a means of generating quantitative predictions for testing the theory. Examples include models of single phenomena (such as Sternberg's model of STM; (Sternberg, 1966)), integrated models covering a wide range of different phenomena (such as Soar (Newell, 1990) and ACT-R (Anderson & Lebière, 1998)), and over-arching principles, which guide the development of models in disparate domains (such as connectionist approaches (McLeod, Plunkett, & Rolls, 1998), or embodied cognition (Pfeifer & Scheier, 1999)).

The group of models to be studied in this tutorial emphasise learning phenomena, and learning at a symbolic level. EPAM was the precursor of the later CHREST (Chunk Hierarchy and REtrieval STructures) system, and both are typically developed from large quantities of naturalistic input. For example, in modelling expert perception of chess players, actual chess games are used (Gobet & Simon, 2000). Similarly, in modelling the acquisition of syntax, large corpora of mother-child interactions are employed to develop the model's long-term memory (Freudenthal, Pine, Aguado-Orea, & Gobet, 2007).

The tutorial is structured so that participants will:

1. Acquire a complete understanding of the EPAM and CHREST approach to computational modelling, and their relation to the chunking and template theories of cognition;
2. Explore some key learning phenomena supporting the chunking theory, based around experiments in verbal-learning, categorisation and the acquisition of expertise;
3. Be introduced to an implementation of CHREST which can be used for constructing models of their own data.

Further information about CHREST, supporting publications and implementations can be found at: `http://chrest.info`

Chunking and Template Theories

A *chunk* is a 'familiar pattern', an item stored in long-term memory. Chunks collect together more basic elements which have strong associations with each other, but weak associations with other elements (Chase & Simon, 1973; Cowan, 2001). Miller observed (Miller, 1956) that short-term memory typically contains a limited number of pieces of information, but the size of these pieces varies with context; this observation lies behind the chunking theory. Chase and Simon (1973) confirmed the presence of chunks in the recall of chess positions, and the EPAM model provides a means of learning, storing and retrieving such chunks.

The chunking theory has been extended to form the *template theory* (Gobet & Simon, 1996, 2000). The extensions include mechanisms to create retrieval structures, which use specific retrieval cues to store and obtain information rapidly. The template is a form of slotted schema, containing a *core*, of stable information, and *slots*, containing variable information. Where the chunking theory captures much of how the average person learns in tasks such as verbal-learning, the template theory further captures the way in which highly-trained human experts perceive and identify patterns in their domain of expertise.

A more detailed overview of the chunking and template theories is contained in Gobet et al. (2001).

Implementation

CHREST comprises three basic modules:

- Input/output module, which is responsible for feature extraction, passing the features to the long-term memory for sorting, and guiding the eye movements;
- Long-term memory, which holds information in the form a discrimination network; and
- Short-term memories, which hold pointers to nodes in the long-term memory.

The key feature which distinguishes EPAM/CHREST models is the discrimination network for storing and retrieving information in long-term memory. Information input to the models is assumed to form a list of subobjects, each of which is either a further list of subobjects or else a primitive. Once information has been stored within the network, it becomes a *chunk*, a 'familiar pattern'. Tests in the discrimination network check for the presence of individual primitive objects, or known chunks (which can be large lists of subobjects). The discrimination network is trained by exposing

CHREST to a large set of naturalistic data. A typical network for an expert in a complex domain will contain on the order of 100,000 nodes.

CHREST extends on EPAM by collecting chunks together when an internal node meets specific criteria relating to its connections with other nodes within memory. A template is then formed from the common information in the linked chunks, with slots created for the variable information. Just as EPAM was the computational embodiment of key aspects of the chunking theory, CHREST implements essential aspects of the template theory.

Input can be provided to CHREST in one of two ways. As a single pattern, which is provided in 'one go'. These patterns are input to the network and stored directly. The second way is to use the in-built attentional mechanism, by which CHREST scans an input array, such as a chess board, and stores parts of the input array into memory. Short-term memory will then hold a set of chunks, each of which may hold information about a different part of the chess board, and collectively holding information about most of the board. The attention mechanism in CHREST is described in Lane, Gobet, and Ll. Smith (2009).

CHREST is implemented in Lisp, and uses Tk to provide a graphical interface. A graphical environment enables users to create simple CHREST models by providing data within an input data file. The implementation also supports more complex tailored models which may be developed by writing special-purpose code using the packages within CHREST. Within the tutorial we will introduce participants to the graphical environment, walk them through a number of provided examples which will illustrate the workings of the architecture and some samples of successful applications, and finally describe the input data format for applying the environment to new domains. A library and manual is provided to assist users wishing to write more complex models.

Applications

The tutorial will cover a variety of experimental data to illustrate the theory and processes. We begin with human verbal-learning processes, which were behind the development of the first EPAM learning system. The interlinked learning operations, which alternately extend or elaborate information in the netwrok, are illustrated using applications in verbal learning (Feigenbaum, 1959; Feigenbaum & Simon, 1984). Further properties of the chunking network will be described with reference to results from categorisation (Gobet, Richman, Staszewski, & Simon, 1997), implicit learning and language learning (Freudenthal et al., 2007; Jones, Gobet, & Pine, 2007).

More elaborate models of expertise explore the interaction between the learner and its external environment. We illustrate this aspect of the theory with models of chess expertise, and in particular look at the recall task, which can reveal many details of expert memory. This application is used to describe CHREST's attention mechanisms (Lane et al., 2009) and how they relate to training the discrimination network.

References

Anderson, J. R., & Lebière, C. (Eds.). (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum.

Chase, W. G., & Simon, H. A. (1973). Perception in chess. *Cognitive Psychology*, *4*, 55-81.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, *24*, 87–114.

Feigenbaum, E. A. (1959). *An information processing theory of verbal learning*. The RAND Corporation Mathematics Division, P-1817.

Feigenbaum, E. A., & Simon, H. A. (1984). EPAM-like models of recognition and learning. *Cognitive Science*, *8*, 305–336.

Freudenthal, D., Pine, J. M., Aguado-Orea, J., & Gobet, F. (2007). Modelling the developmental patterning of finiteness marking in English, Dutch, German and Spanish using MOSAIC. *Cognitive Science*, *31*, 311–341.

Gobet, F., Lane, P. C. R., Croker, S. J., Cheng, P. C.-H., Jones, G., Oliver, I., et al. (2001). Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, *5*, 236–243.

Gobet, F., Richman, H., Staszewski, J., & Simon, H. A. (1997). Goals, representations, and strategies in a concept attainment task: The EPAM model. *The Psychology of Learning and Motivation*, *37*, 265–290.

Gobet, F., & Simon, H. A. (1996). Templates in chess memory: A mechanism for recalling several boards. *Cognitive Psychology*, *31*, 1–40.

Gobet, F., & Simon, H. A. (2000). Five seconds or sixty? Presentation time in expert memory. *Cognitive Science*, *24*, 651–82.

Jones, G. A., Gobet, F., & Pine, J. M. (2007). Linking working memory and long-term memory: A computational model of the learning of new words. *Developmental Science*, *10*, 853–873.

Lane, P. C. R., Gobet, F., & Ll. Smith, R. (2009). Attention mechanisms in the CHREST cognitive architecture. In L. Paletta & J. K. Tsotsos (Eds.), *Proceedings of the fifth international workshop on attention in cognitive science* (Vol. LNAI 5395, pp. 183–196). Berlin: Springer-Verlag, GmbH.

McLeod, P., Plunkett, K., & Rolls, E. T. (1998). *Introduction to connectionist modelling of cognitive processes*. Oxford, UK: Oxford University Press.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*, 81–97.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Pfeifer, R., & Scheier, C. (1999). *Understanding intelligence*. MIT Press.

Sternberg, S. (1966). High speed scanning in human memory. *Science*, *153*, 652–654.

A Summary of *Human-System Integration in the System Development Process*

Frank E. Ritter (frank.ritter@psu.edu)
College of Information Sciences and Technology
Penn State
University Park, PA 16802

Abstract

In a recent book, Pew and Mavor and the Committee on Human-System Design Support for Changing Technology (2007) proposed a revision to Boehm's Spiral Model for system development. This revision encourages considering the user within a system as a source of risk. Where these risks are significant, this approach suggests ways to reduce the risk through appropriate studies of the user. This tutorial provides a summary of this model and some of the insights and extensions of this model based on teaching it. These insights are related to learning: learning by the field through using this approach to organize methods and techniques, learning by system development managers that there are sometimes risks related to humans using their systems (and implications for how to teach this), learning about designers as stakeholders, and learning by designers as lessons from one design are applied to later designs. These insights and extensions suggest the importance of shared representations such as cognitive models for educating team members and for the system development process.

Keywords: Human-system design; user models; representation

Introduction

In a recent book, Pew and Mavor and the Committee for Committee on Human-System Design Support for Changing Technology (2007) propose a revision to Boehm's Spiral Model for system development. I present here a summary of this model for system design. This report argues that not understanding aspects of the user can be a risk in system design. Thus, where there are no user related risks, system designers do not need to worry about users. In other cases, where there are risks, the book presents approaches for reducing these risks. User models are a way to share knowledge about users across the design process.

Intended audience. This tutorial will be of interest to people interested in using models in industry as a shared representation, modelers interested in applications of models, and those interested in understanding the Committee's report as edited by Pew and Mavor.

Prerequisite knowledge: This tutorial does not presume any prerequisite knowledge. Attendees may wish to have skimmed the book (which is available on the web page-at-a-time for free), or have examined other work on system design.

The Spiral Model

The spiral model is an approach to system design that encourages increment development of systems in a spiral of requirements specification, technical exploration, and stakeholder commitment. The spiral model is shown in Figure 1, where movement around the spiral represents time and commitment and work on the project.

At each stage in development, the system development is accessed for risks to the system's success. The process is then targeted at reducing these risks. Some risks may be technical, can we build it or can we build it for that price? In these cases, technical work is performed to reduce the risk through technical understanding. Other risks can arise from historical events, which are hard to reduce, and from financial matters, which often can be reduced by setting up contracts at a known price. Risks can also occur due to not understanding user, their tasks, or their interaction with the system, which the report and this tutorial address.

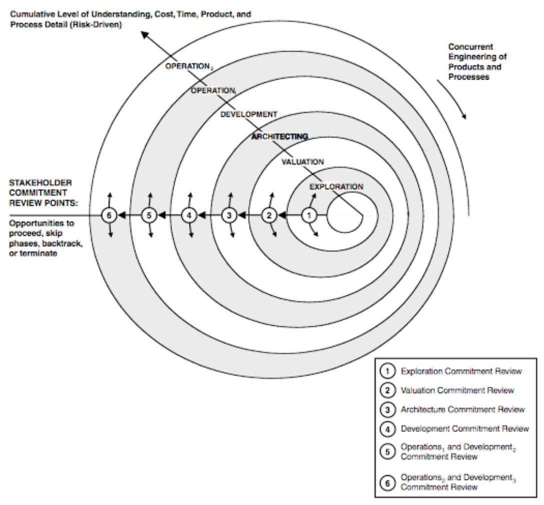
Cumulative Level of Understanding, Cost, Time, Product, and Process Detail (Risk-Driven)
Concurrent Engineering of Products and Processes
OPERATION$_2$
OPERATION$_1$
DEVELOPMENT
ARCHITECTING
VALUATION
EXPLORATION
STAKEHOLDER COMMITMENT REVIEW POINTS:
Opportunities to proceed, skip phases, backtrack, or terminate
6
5
4
3
2
1
1 Exploration Commitment Review
2 Valuation Commitment Review
3 Architecture Commitment Review
4 Development Commitment Review
5 Operations$_1$ and Development$_2$ Commitment Review
6 Operations$_2$ and Development$_3$ Commitment Review

Figure 1. The basic spiral model (Pew & Mavor, 2007).

This revised system design model in Pew and Mavor (2007) has several key features, as noted in the book: (a) Systems should be developed through a process that considers and statisfices the needs of stakeholders. This step is done in the Exploration and Evaluation stages.

(b) Development is incremental and performed iteratively. These related aspects are shown in Figure 1 by the multiple loops representing the increasing resources committed to design and implementation, and through the five stages (Exploration, Valuation, Architecting, Development, and Operation). These stages are incremental because movement from one stage to another depends upon a successful review

(c) Development occurs concurrently, that is, multiple steps may be performed simultaneously. Designers may implement one part of the system while testing another.

(d) The process is mindful of risks during system development and deployment. The level of risk is accessed repeatedly at milestones between stages. Risk is used to manage the project—the level of effort and level of detail of work are driven by the level of risk. Where there is no risk to system development, there is no need for effort to reduce risk. For example, if the system being developed is similar to a known product, there may be no reason to explore further how to support users or how to manufacture it.

Insights

The committee did not set out to create human-system integration (HSI) teaching materials, but the resulting book can be used to teach about HSI, human-computer interaction (HCI), and human factors. In teaching this material, the students and I found several extensions and insights.

(a) The revised spiral model provides a framework for organizing much of HCI and HSI. Most HCI methods can be cast as ways to reduce various types of risks, and most design processes cast as steps in the spiral.

(b) The revised spiral model is not just normative, it is also descriptive. That is, managers may already be working to reduce risk; it is just that they do not see the risks related to users because they do not understand users. This insight suggests that it is likely to be more important to create materials to teach about incipient risks than it is to teach about the revised spiral model process itself.

(c) Designers are stakeholders too. Tools and approaches to reduce risks must support their understanding. They are users of the process and their needs and capabilities are part of the development process.

(d) One of the major results of using shared representations and analyses of systems while being designed may be learning of the design team and application to later designs. Thus, work on creating shared representations should not just include integration across the team and across the design process for a single project (which the book calls for), but also across designs over multiple projects.

Summary

The risk-driven incremental concurrent development model, the later version of the spiral model, provides a useful and safer way to create systems. As a study aid, the model provides a new way to view HSI and HCI methods, design approaches, and development theories, and how to include them in system design.

So, in this new view, the decision to do user research, review, or studies is based on system design risks. If the system development is predicted to be smooth and not novel, then little or no usability studies are required, and little or no should be done. Where there is more risk, more work should be done given the resources. But, the user-related risk has be balanced against other risks. The technology may in fact be riskier, and thus require more resources. Or, as is often the case, the managers understand the technical risk.

There are several corollaries to this. The managers often must be educated about user risks, and we will need books and tutorials like this to help educate system designers about where and when their theories of users mismatch the world.

We will need improved representations of users (shared representations) to use in the design process, similar to how blueprints are used in buildings.

Acknowledgements

A graduate class on HCI at Penn State and a group at QinetiQ Malvern provided useful discussions that lead to the preparation of this tutorial and provided useful comments. Barry Boehm provided some slides and figures that I used to create my teaching materials. Olivier Georgeon, Jon Morgan, and Rick Koubek provided comments that improved the presentation.

References

Pew, R. W., & Mavor, A. S. (Eds.). (2007). *Human-system integration in the system development process: A new look*. Washington, DC: National Academy Press. books.nap.edu/catalog.php?record_id=11893.

Frank Ritter currently teaches at the College of IST at Penn State, previously he has worked at BBN Labs and taught at the U. of Nottingham, where he was the Director of the Institute for Applied Cognitive Science. He earned his PhD in AI and Psychology and a MS in psychology from Carnegie-Mellon. He has a BSEE from the University of Illinois at Urbana/Champaign. He was a member of the committee that helped prepare the book the tutorial is based on.£