Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Adaptivity: A Quality Goal for Agent-Oriented Models?

Leon Sterling

Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn, Victoria, 3122, Australia (e-mail: lsterling@swin.edu.au)

Abstract: If software systems are to behave more intelligently, they will need to be able to adapt to the changing external world. The agent-oriented paradigm offers the potential to build adaptive systems. This paper explores the nature of adaptivity in multi-agent systems by exploring how to represent adaptivity in agent-oriented models. Rather than focussing on mechanisms for individual agents to be highly adaptive, we advocate regarding adaptivity as a quality goal of an overall multi-agent system. Discussing agent-oriented models should help guide where design tradeoffs need to be made. Learning can be approached in a similar way to adaptivity, where learning is viewed as an attribute of the system rather than a fundamental property of individual agents. Examples are given from use of automated teller machines at mobile sites and intelligent information agents.

Keywords: agents, modelling, adaptive systems, quality goals.

1. INTRODUCTION

A challenge for engineers is how to design systems that work effectively in today's complicated world. Five key characteristics of modern computing environments that need to be taken into account when building systems are: complexity, their distributed nature, time-sensitivity of information, the uncertainty and unpredictability of the surrounding environment, and its open nature – new things happen (Sterling and Taveter, 2009). This is sometimes summarised as systems being dynamic.

A desirable attribute for software to be part of a dynamic system is *adaptivity*. As the world changes, our software ideally should evolve to reflect the change. As new facts enter the world, the software should not break, but absorb the new facts and change behaviour as appropriate. An obvious area where adaptivity is essential is security. As a new virus or security threat is determined, it would be good if the system which checks viruses and is responsible for firewall security system could incorporate a response to the new threat automatically.

How does a system know how to adapt in response to new events and situations? There has to be some purposefulness that the system can refer to. To refer to software security, the system needs to recognise that is should respond to all threats not just the ones explicitly listed. The overall system purpose should be factored into the system design so that the software will be able to change its behaviour. Furthermore, the software change and adaptation should be understandable. Relatedly, we need ways of thinking about and describing software which simplifies complexity, at least in regards to describing behaviour.

How do you design software to allow such adaptivity? I claim that both an appropriate metaphor is needed along with good conceptual tools. This paper argues that agent-oriented modeling has an important contribution to make, building on the agent-oriented models and conceptual framework laid out in Sterling and Taveter (2009).

This paper is organized as follows. Section 2 briefly examines both models and agents, and how they combine in agent-oriented modeling. Section 3 discusses quality goals and introduces how adaptivity may be conceived as a quality goal. Section 4 sketches how different conceptions of adaptivity may affect system development, using as a context mobile automated teller machines. Section 5 then discusses how to treat learning in multi-agent systems in a similar manner to adaptivity, and gives an example in the context of an information agent finding places to live. Section 6 gives some related work and is followed by conclusions.

2. AGENT-ORIENTED MODELLING

Let us consider 'what is a model?' A definition taken from the Web is that a model is a *hypothetical description of a complex entity or process*. A model is constructed to aid in building the intended system. Modelling is empowering in a practical sense. If you can model, you are a significant part of the way to building something useful. Furthermore, the models must have sufficient detail to be useful, but not too much detail as to overwhelm.

As for 'what is an agent?', computer science researchers have used the word agent for over twenty five years with a range of different meanings. For the purpose of this paper, an agent can be regarded as *an entity that performs a specific activity in an environment of which it is aware and can respond to changes in the environment*.

One obvious consequence of the informal definition of agent that is worth explicitly pointing out, is that people are agents. People live in the world, are aware of changes in the world of many different factors and attributes including weather,

politics, and social organisations. People act in the world. For example, to look at the changes mentioned, they may protect against the weather by carrying an umbrella, putting on sun screen lotion or a snow suit, though usually not simultaneously, vote in an election to influence politics, and form networks of friends. Being agents helps people to conceptualise and understand systems in terms of agents.

Individual agents can be interesting. Interactions between agents can also be interesting, as are interactions between an agent and the environment in which it is situated. Adopting the agent metaphor for developing software is raising both the visibility and abstraction level of interactions between agents. To appreciate the value in being able to understand agents and their interactions, we need to consider a broader systems view. Loosely, a system is a set of entities or components connected together to make a complex whole or perform a complex function. Where several, perhaps all, of the connected entities are agents, we have a multi-agent system.

The previous section argued that adaptive multi-agent systems are a natural approach to cope with a dynamic world. The question then arises: How do we design adaptive multi-agent systems? This paper advocates using agent-oriented models based on the conceptual framework discussed in Sterling and Taveter (2009).

The framework contains models at three levels of abstraction: loosely motivational models, design models, and implementation models.

Table 1 lays out the range of models envisaged. It is beyond the scope of the paper to discuss them all in depth. Rather we will give some examples which give the flavour of the modelling exercise.

TABLE I. THE MODEL TYPES OF AGENT-ORIENTED MODELLING

	Viewpoint aspect		
Abstraction layer	Interaction	Information	Behaviour
Motivation	Role models and organization model	Domain model	Goal models and motivational scenarios
Design	Agent models, acquaintance model, and interaction models	Knowledge model	Scenarios and behaviour models
Implementation	Platform-specific design models		

To illustrate the most accessible of the models, the figure below gives an example of a goal model for handling potential intruders in a house. The example has been widely used in teaching at the University of Melbourne to explain agent-oriented thinking, and is discussed in detail in Chapter 9 of 'The Art of Agent-Oriented Modelling' (Sterling and Taveter, 2009).

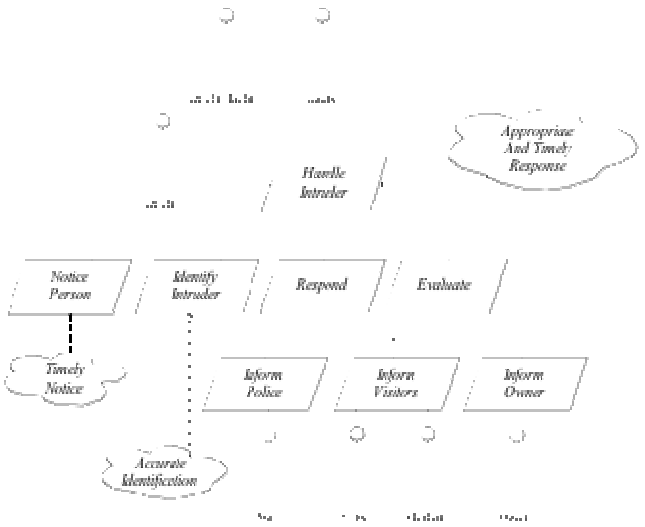

FIGURE 1. HIGH-LEVEL GOAL MODEL OF THE INTRUDER SCENARIO

The figure contains three modelling entities, parallelograms which represent goals, or elements of functionality that the system will deliver, stick figures which represent roles which abstract the capabilities for undertaking the functionality, and clouds which represent quality goals which are attributes the system needs to have.

The overall goal of the system is to handle intruders. This top level goal is decomposed into four subgoals: noticing that there is a potential intruder in the house, identifying the person (probably by capturing a picture and checking it against some data base), respond to the presence of the person, and evaluating the handling of the incident. The response is decomposed into three subgoals: notifying the police, notifying potential visitors to the house, and informing the owner. There are seven roles involved: the intruder, the intruder handler, the owner, the police, visitors, the scheduler of the visitors and the overall evaluator. The overall quality goals are that the noticing of the intruder be timely, the identification be accurate and the overall response be timely.

3. QUALITY GOALS

In implementing a multi-agent system, motivational goal models are translated into design models for agents such as agent models and interaction models. These express the activities that an agent will perform in fulfilling its responsibilities, and the interactions that the agent will have with other agents in the system. In designing the interactions, one needs to understand where adaptivity can arise.

At first thought, one might be tempted to place the capability for adaptivity within individual agents. Such thinking has led many researchers to look for general theories of adaptive agents. My contention is that adaptivity is rather a quality goal of the overall system and needs to be part of overall system design. This is the core idea of this paper.

In this section, we explore *quality goals* within agent-oriented modelling following the approach of Sterling and Taveter (2009). Quality can mean several things, ranging through excellence, fitness for purpose and a general attribute. We take the last sense and define a quality goal as one which specifies an attribute of a system, such as ease of use or security.

Focussing on quality is well established within software and systems engineering. Software engineers are aware of the need to express quality attributes of software as well as functional capabilities of software. These quality attributes are referred to using a variety of terms including: non-functional requirements, constraints, quality attributes, quality goals, or quality of service requirements (Sommerville, 2007).

Some quality goals can be quantified. For example, drawing on the figure above presented for intruder handling, we can specify what the quality goal means for recognising that an intruder is present be timely. A possible quantification is that the system component responsible for noticing the intruder needs to send an alert within two seconds of the intrusion. This will place constraints on the system.

Other quality goals are not so easily quantified. Another project we have been involved in, reported in (Paay et al., 2009), involves designing technology to allow grandparents and grandchildren to interact over distance in a fun way. A goal model from the project is given below.

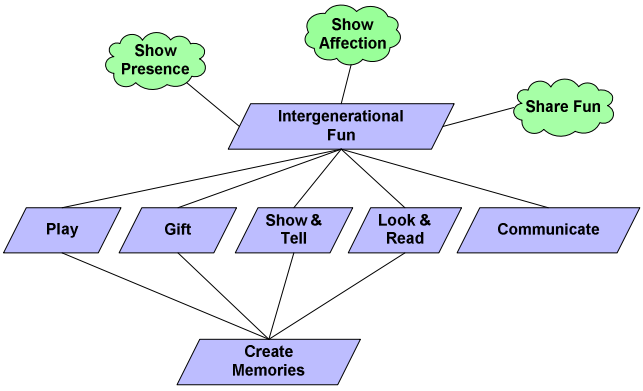

FIGURE 2. GOAL MODEL OF INTERGENERATIONAL FUN

There are three quality goals, none of which are easily quantifiable. Grandparents and grandchildren sharing fun will vary considerably from family to family, and therefore ethnographic techniques were involved in assessing what fun might constitute in a range of families. Clearly, also, sharing affection which is an affective property, needs to be assessed at a system level and not just at the level of an individual agent.

We maintain that there is benefit in articulating quality goals without the need to resolve them into measurable goals. Our agent-oriented models allow the expression of non-functional requirements by attaching quality goals to goal models. There is a direct pairing between system goals and quality goals, whereas non-functional goals do not generally have a direct relationship with functional goals (Chung et al., 2000). This makes it more difficult to carry them through the process in an unresolved state. Relating an abstract and unresolved quality attribute to a system goal enables a focus on a broader set of quality goals within the design process.

By capturing and representing quality goals in agent-oriented models we make a commitment to important aspects of social interactions that can remain unresolved, giving interaction designers and software engineers alike a focal concept for analysing and designing around complex social concepts. By externalizing them in a simple format the models become shared artifacts (Paay et al., 2009) that are able to sustain multiple interpretations across disciplines. Quality goals allow a focus on understanding the reasons why people do things, or the essence of a relationship rather than describing a physical action. In doing so, quality goals capture something that is more dynamic and fluid than other elicitation mechanisms found in usual software engineering practices.

Loosely, quality goals are considered at three levels. At the motivational level, they are abstract aspirations for the system. In our experience they are useful remaining abstract to help shape conversation and encourage engagement between the various stakeholders.

During the design stage, the design team is encouraged to think about design trade-offs. Specifically how the design trade-offs will affect the overall quality goals and outcomes/objectives of system use. The design team is able to make appropriate decisions.

The implementation team will know of specific techniques that will help accomplish the design objectives. It is at the implementation level that performance constraints will need to be met, but whether that is possible will depend on the design. Cryptographic techniques to increase software security are also at the implementation level and again need to be considered in the context of the design and whether they are appropriate for the overall system level objectives of security. Research over the past decade indicates that quite varying quality goals can be handled in this manner. Standard quality goals such as performance and security, engineering goals as safety, and abstract goals such as having fun, sustainability, and having respect for others can all be treated in an analogous manner.

This paper advocates to view adaptivity as a quality goal. The idea is to have adaptivity as an overall system objective. During design, tasks, protocols and interactions will be prescribed and analysed to see where adaptivity can be achieved. Individual mechanisms will be put in place during implementation to meet the design objectives.

4. DEVELOPING MOBILE ATMS

So far we have discussed agent-oriented modelling and quality goals, and advocates viewing adaptivity as one. To make the discussion more concrete, let us consider how adaptivity might play out in a specific example. The example

arises from discussions with a colleague, Geoff Jenkins, who is currently testing software to facilitate mobile banking at public events. The idea is to provide mobile automated teller machines (ATM) at venues such as sporting events with large crowds, entertainment venues, etc. The ATMs need to communicate with customers and with banks.

The system can clearly be conceived as a multi-agent system. The bank, customers and the ATM can all be considered as agents. Understanding the interactions and designing for them is key to building a successful system.

Because of the lack of knowledge of specific venues there have to be decisions about access. In the actual implementation there have been issues with the communication signals and what to do if a signal drops out. Technically, on mobile networks, a stationary client can be easily 'orphaned' if the carrier resets its network registration status. In this case, because the client does not move, it does not change cell and so will never, ever re-register.

There is intelligence involved in knowing whether signals are alive or not, and when to restart some of the services to right any problems that may have arisen. Since the resets usually happen overnight, software on the client must always force a reboot early each morning. The time at which this should happen needs to be optimally late. There is a definite need for adaptivity to handle such issues as changing the refresh/restart rate, updating the modem status needed, and determining frequency of monitoring.

Taking the adaptive agent approach would put all the responsibility for behavioural intelligence into the agent that communicates from the ATM to the satellite or other communication channel and onto the bank. That would be difficult. Taking the adaptive system approach would analyse each of the interactions. The design had identified the various protocols. The protocols can then be studied to see where adaptivity may be required. Algorithms can then be developed to improve the interactions by monitoring and adjusting the availability of the various components of the system. This latter approach simplifies life for the developer and better suggests a pragmatic approach to what needs to be done.

5. LEARNING

There has been a lot of consideration of the role of learning within multi-agent systems. It has even been claimed that learning is a fundamental characteristics of agents. Insects can demonstrate quite flexible behaviour without learning, and perhaps they should be considered as agents. Discussing such a topic in depth is beyond the scope of this paper.

In my experience, learning is less needed in practice. Indeed industrial practitioners are often scared by the thought of an agent autonomously learning and changing behaviours (Munroe, et al., 2006).

Let us see how learning can be treated in the same manner as adaptivity. Focussing on learning itself means that individual learning agents are required. There has been considerable work, though it is often hard to apply to new domains. Treating learning as a quality goal of the system involves analysing the interactions during design and seeing what learning techniques may be applicable.

Consider an intelligent system for helping a new arrival settling in a new city and specifically the component that may help the new arrival find long-term accommodation. Such a system was prototyped in the Intelligent Agent Laboratory in the University of Melbourne over ten years ago by a PhD student, Sharon Gao, under the author's supervision (Gao, 2000). The source of the information was classified advertisements in newspapers and online Web sites. While there are now consolidated Web applications that help in such a task, the Web was not as widely useful when the system was designed and prototyped.

One problem that the new arrival has is knowing the suburb names in the new city and where the suburbs are in the city. That is a particular problem in Melbourne which is spread out over a broad geography. The new arrival, supported by the system, needs to learn many things, the geography of the city, the going rates for flats and houses, and what is good value for money and meets the constraints of the new arrival. The system was envisaged as building up the appropriate knowledge from scanning the newspapers.

The prototype that was exhibited showed two forms of learning. It learned suburb names using a mixture of knowledge of syntax and comparing the format of several ads. It also learned what the average cost of flats was and when an advertised rate seemed cheaper than average. The net effect was successful, though not extensively tested.

The method that was used by the information agent was not particularly sophisticated and could not be considered a learning agent. However the system overall did learn in the standard sense of improving its behaviour over time. It is also not clear that any more sophisticated approach to learning would have been justified for this particular application.
The system could further learn by adding in new methods for other interactions. The new methods can be added without a deep insight into mechanisms of learning. Nevertheless the system as a whole improves its behaviour.

6. RELATED WORK and CONCLUSIONS

There is considerable literature on incorporating adaptivity in software systems. One approach has been to define adaptive objects. A Google search on "adaptive objects software" revealed (Liberherr, 1995) as its top hit among eleven million hits. A Google search on "adaptive agents software" gives over eight million hits with the top returns pointing to articles written in the 1990's. For all of the attention, there has been no predominant method.

There has been also much research on agent-oriented software engineering, of which agent-oriented modelling is a part. A modelling approach for bottom up development of

adaptive multi-agent systems is discussed in this proceedings by Taveter and Kirikal (2011).

The approach to quality goals built on the work of Chung et al., 2000, but has chosen a more holistic, qualitative approach rather than a quantitative approach. My own research continues to investigate nonstandard quality goals, such as being engaging, or developing cool software in collaboration within the Faculty of Design at Swinburne University of Technology. A discussion of sustainability as a quality goal of systems based on some early work is presented in Pedell and Sterling (2011).

In conclusion, this paper has advocated the position that in order to develop an adaptive multi-agent system, one should view adaptivity as a quality goal. Developers do not need to solve the problem of what is an adaptive agent and struggle with a comprehensive theory of adaptive agents and what mechanisms are needed to make agents adaptive. Rather, adaptivity should be regarded as a system-wide characteristic to be considered when designing the multi-agent system and building the architecture. Adaptive behaviours that are needed for the system are built in accordingly. The approach was discussed in the context of a portable system for automatic teller machines for dispersing money at sporting events.

A similar kind of analysis can be undertaken for other system characteristics. We demonstrated the approach for learning. An intelligent information system for learning about accommodation in a new city was presented in this way. The approach has also been addressed for sustainability.

Our future work is to continue to investigate how to embed nonstandard quality goals into software engineering practices. We are working with digital media designers on making interactive games engaging, for example. There is another plausible line of research that may view adaptivity as an emergent property.

ACKNOWLEDGEMENTS

The author would like to thank Dr. Geoff Jenkins for helpful discussions. The author would also like to thank various members of the Agent Research groups at the University of Melbourne and Swinburne University of Technology, including Tim Miller and Sonja Pedell. This work was supported by Australian Research Council Grant LP0882140.

REFERENCES

Chung LK, Nixon BA, Yu E, Mylopoulos J (2000) *Non-Functional Requirements in Software Engineering*. Kluwer Publishing

Gao, X. Knowledge-based Information Extraction from the World Wide Web, PhD thesis, University of Melbourne, 2000

Liberherr, K . *Adaptive Object-Oriented Software: The Demeter Method with Propagation Patterns*, PWS Publishing Co. Boston, MA, USA 1995

Munroe, S., Miller, T., Belecheanu, R., Pechoucek, M, McBurney, P. and Luck, M. Crossing the Agent Technology Chasm, Lessons, experiences and challenges in commercial application of agents, *Knowledge Engineering Review*, 2006, 21: 345-392

Paay, J., Sterling, L., Vetere, F., Howard, S., and Boettcher, A. Engineering the Social: The Role of Shared Artifacts, IJHCS, 2009, 67(5):437-454.

Pedell, S., and Sterling, L., The Benefits of Agent-based Motivation Models in Policy Formulation and Implementation. V. Dignum (Ed), Proceedings of the 1st Workshop on Agentbased Modeling for Policy Engineering at AAMAS, 2011

Sommerville, I. *Software Engineering* (8th edition). Addison Wesley, Harlow, Essex. 2007.

Sterling, L. & Taveter, K. (2009). *The Art of Agent-Oriented Modeling*. Cambridge, MA, and London, England: MIT Press.

Taveter, K. and Kirikal, K. (2011). Designing Adaptive Multi-agent Systems by Agent-Oriented Modelling. This volume