Reflections on Model-Based Design: Definitions and Challenges

Stephanie Wilson

List of participants

Mark Addison, Tom Bösser, Con Copas, Peter Forbrig, Andreas Homrighausen, Frank Lonczewski, Josef Voss, Stephanie Wilson, Takayuki Yamaoka.

Abstract

This paper reports a working group discussion addressing various issues pertaining to model-based design raised at the CADUI'96 workshop. Since the term 'model-based design' was first applied in the context of interactive system design its usage has been broadened beyond the original definition to include a wide range of design approaches that involve modelling activities. Therefore, a key question for the nine participants of the working group was what constitutes model-based design? The working group further reflected on the current state of the art in model-based design, the limitations of the techniques and challenges for the future.

Introduction

The CADUI'96 workshop offered a timely opportunity to review progress in the field of model-based design, to examine the current state of the art and to look to future challenges. This paper reports the deliberations of one working group convened during the workshop which reflected on these issues.

The nine participants in this working group shared common interests in model-based design, automatic generation and task modelling, although they represented a variety of backgrounds (industrial / academic, software engineering / psychology). The working group was charged with addressing three discussion points:

- What models could or should be used in model-based design?
- How can the models be used at run-time?
- What are the limits / problems of model-based design?

The discussion on each of these points is reported in sections 2, 3 and 4. However, much of the available time was spent establishing a common ground for the discussion and reflecting on issues raised by the workshop so far. One important issue for the group was the question of what constitutes model-based design; this is reported in section 1 as it provides the basis for the remainder of the discussion. Some general concerns were also voiced by the group.

For example, the observation that while a plethora of model-based techniques have been reported recently in the research literature, there are few reports of industrial application. Moreover, many of the systems are markedly similar in their capabilities, suggesting that few advances are being made and that there is perhaps a tendency to repeat the same mistakes in different systems.

It should be noted that due to timing constraints, the working group had only one opportunity for discussion during the CADUI'96 workshop. This report has been compiled from that discussion and subsequent contributions made by the participants.

1 What constitutes model-based design?

As a preliminary to considering the models that could or should contribute to design, a significant part of the discussion was devoted to addressing the question of what constitutes model-based design. Participants had different interpretations of the term model-based design. There was general agreement that the term originated from domain modelling and problem solving literature in the AI community. Systems such as UIDE [Foley91] were probably the first to coin the term 'model-based' in the context of user interface design, and had a clear connection to the use of the term in AI. These systems incorporated declarative models, inference engines and problem solving techniques. However, many of the approaches and systems presented at the CADUI'96 workshop would characterise themselves as model-based, in spite of the fact that they make no use of problem solving techniques.

This led into a discussion of what is a model, where there were two distinct views. One view held that models are abstract declarative representations of real world entities that can be used for reasoning, and that therefore a design technique is only model-based if there is a tool providing some level of automation or reasoning. The alternative view held that anything which provides an abstract representation of some information may be regarded as a model, and that any design process based around models could be termed model-based, irrespective of whether or not the models are used for reasoning. The latter view appears to reflect the commonly adopted terminology in the HCI community at present.

This currently accepted understanding of model-based development can be summarised by this characterisation offered by one participant, Josef Voss:

- model-based development works with a set of related models tailored to the problem domain in general and the project under consideration;
- models are fixed points in the development process, guiding the progress from abstract / user-centred concepts to system realisation;
- the development process itself can take various paths between these fixed points;
- each model has a certain level of abstraction and provides a certain view of the project.

A final discussion point on the theme of what is a model was "what is a task model"? Again, there were a number of different viewpoints, highlighting the fact that people from different backgrounds had rather different perspectives.

Some comments were made concerning the appropriate level of abstraction for a task model: should it describe the work that people do or should it give a detailed account of their interaction with a particular system? This led to the question of whether or not a task model can be independent of the technology.

The generally held view was that, at some level, a task model can be independent of any specific interactive system, in the sense that it can be independent of the specifics of presentation details and of low-level interaction with particular widgets. Other questions concerned whether hierarchy and sequencing should be expressed in a task model, and, if so, how should they be represented.

2 What models could or should be used in design?

The discussion on this point attempted to understand what aspects of a design situation should be modelled during the design process. Given the time constraints, the discussion was somewhat inconclusive and it was only possible to touch upon some of the models that have contributed, or might contribute, to a model-based approach. These included:

- problem domain models (including domain object models);
- task models (of existing and envisioned user tasks);
- user models;
- interaction models (at different levels of abstraction and including information about dialogue and user interface components);
- models of design knowledge;
- implementation platform models.

These models do not represent all the information that is used during the design process, nor are they ever likely to do so. The important point is that the models should make explicit, and focus attention on, important information that might otherwise be overlooked, and that they should do so in a manner which facilitates the designer making use of the information in the creation of cost-effective and usable design solutions.

3 How can the models be used at run-time?

Current model-based techniques are largely concerned with supporting interface developers in the creation and realisation of interactive system designs. They are intended for design-time use. Run-time use, on the other hand, means moving towards supporting the users in their interaction with the resulting systems. Of course, the fact that design models have the potential for run-time use does not mean that it is necessarily a good idea to use them in this way. Further research is required to determine how effective the design models might be in supporting the users at run-time.

Many of the opportunities for run-time use rely on the fact that models created during the design activities persist in machine-readable form in the run-time environment. (However, there are also other 'run-time' uses for the models. For example, a task model might be used as the basis for producing a training manual.) In order to make use of design models at run-time, the run-time system must track and maintain references between models. References in both directions are useful: up-stream references from system-level models to user-level models, e.g. from a button to a task description, and downstream references from user-level models to system-level models, e.g. from a task action to a button. This would, for example, allow help to be supported in both directions, "What does this button do?" and "How can I accomplish this task?".

Different models offer different possibilities for run-time use: high-level models such as task or domain models can be exploited in the provision of powerful help systems. A user model can offer information, not just about user preferences, but about forms of interaction or representations that might be appropriate for a given user population performing a particular task. Lower-level models can support the user by explaining the structure of the system itself and can provide a basis for user modification /configuration of the system. For example, the interaction model could be interpreted at run-time, facilitating configuration by users, or it could be regenerated interactively at run-time, allowing modifications to be made at the level of the task, user or problem domain models.

4 What are the problems / limitations of model-based design?

There were two main thrusts to the discussion on this point. Firstly, the problems and limitations of the model-based approaches in general were considered, and, secondly, those of automatic generation in particular were addressed. Many of the opinions expressed by participants in the working group reflected views voiced elsewhere during the workshop, notably by Pedro Szekely during his plenary presentation [Szekely96].

There was general consensus that model-based techniques have not, as yet, lived up to the expectations and claims of their proponents by demonstrating their value

in practice. It was suggested that we can only start to examine specific limitations of model-based techniques after detailed study of realistic applications, of which there are remarkably few at present. Certain data-centred approaches have been successful in generating business-oriented applications where the emphasis is on visualising and/or modifying form-based data, but the general worth of model-based techniques has not been demonstrated for other types of interactive systems, such as professional systems (e.g., medical equipment).

Some participants held the view that model-based design, and specifically model-based generation, of user interfaces works best in restricted application domains and for precisely defined work procedures such as the typical form-filling interfaces for transaction processing mentioned above. Others questioned this view, believing that while this may reflect the current state of the art, it is not an inherent limitation of the approach.

Not surprisingly, a second point of discussion regarding the problems of model-based design focused on the problems and limitations of automatic generation. This had proved to be a contentious issue throughout the workshop. Some participants were wholly convinced of the merits of the idea and were keen to incorporate as much automation as possible into systems, with increasingly sophisticated generator tools and complex design guidelines.

They offered arguments such as that this approach meant more rapid application development times (by reducing the designer's workload), that it guaranteed the generated system would meet some minimum standards, that it resulted in consistency across user interfaces, etc. Other participants were less convinced, believing that automatic generation is a difficult problem and that it is therefore unrealistic to expect any automated tool to ever be good enough. They cited reasons such as the difficulties in coping with the diversity of application domains and the potential lack of innovation or novelty in the generated design solutions (while automation might guard against bad design solutions, it was thought unlikely to result in the 'best' solutions).

Further arguments centred on the observation that by the time design knowledge has been assimilated and embodied in a set of sophisticated design guidelines, the user interfaces generated by these tools tend to lag a generation behind current developments. While model-based techniques are currently approaching the point where it is possible to generate limited WIMP type interfaces, current technology has moved forward to multimedia and virtual reality systems.

It was felt that some form of design assistance, rather than full automation, could be an alternative avenue to explore for supporting interactive system design. In this scenario, automatic generation and manual development would be combined so as to complement each other. For example, a transformation step starting with only a part of the source model could be used to extend or complete a manually created model. Automatic generation should be used primarily for activities that are tedious to perform manually and are well understood (so that a body of design knowledge

exists to guide the automation). For example, it could help with low-level prototyping or implementation activities, it could provide default translations between different models, or it could assist in model visualisation.

Finally, it should be noted that different forms of automatic generation are possible, some of which may be more or less feasible in practice and in their acceptability to the design community. The term is usually taken to mean either the generation of abstract interaction models from task, user and problem domain models, or the generation of concrete user interfaces from abstract interaction models and design guidelines. However, automatic generation can be used to produce resources other than the interface itself. For example, task models could be used to generate evaluation scenarios and test cases, while task models and abstract interaction models could contribute to the generation of help systems.

5 Summary and future challenges

Within the working group there was some sense of reflecting on the state of the art in model-based user interface design and on the future challenges for research and development work in this field. To date, researchers have demonstrated that model-based techniques can support the design of interactive systems and have shown how models capturing various forms of design information can contribute to such a design process. They have also provided numerous examples of software tools to support these techniques, for example, tools to support the construction of models, reasoning about models, or the generation of models.

The immediate challenge for the model-based design community is to offer evidence of the practicality of these techniques. This requires the application of the techniques to real world design problems and a more rigorous assessment of their strengths and weaknesses in such use. In particular, we need to examine the validity of claims such as "model-based techniques offer a cost effective approach to the design of usable systems". How do the costs of a model-based approach compare with those of other design techniques, and how effective are the resulting designs? We need also to examine whether these techniques, and their supporting tools, are capable of delivering systems in the technologies of today rather than yesterday.

Modelling is frequently a time-consuming, and therefore expensive, activity. The added value of choosing to model certain information explicitly during design is likely to vary between one design situation and another. Therefore, there are questions to be asked concerning the costs and benefits of modelling information explicitly, as opposed to leaving it implicit in the design context. For example, what, if any, is the added value of using explicit user models? Similar questions must be asked of the tools. We need to determine where software tools are the most effective approach to supporting design-time modelling activities, and where other approaches might be more appropriate (e.g., paper-based models). Likewise, we need to investigate where model-based tool support might genuinely enhance the user's interaction with a system at run-time.

The final challenge discussed by the working group lies in the area of automatic generation. Can we reconcile the two opposing schools of thought evident at the CADUI'96 workshop, with one party eager to increase the level and sophistication of automation, while others believed that, at some level, design decisions are best left in the hands of the designer? The group felt that some compromise may offer the best solution by taking advantage of the strengths of each approach.