21

Mini-workshop: how to make formal methods useful

Chr. Gram
Technical University of Denmark
cg@id.dtu.dk

G. Abowd
Georgia Institute of Technology
abowd@cc.gatech.edu

Participants

Participants were: Gregory Abowd, Christian Gram, Balachander Krisnamurthy, Reino Kurki-Suonio, Ole Lauridsen, Philippe Palanque, Fabio Paterno, Manfred Paul, Joergen Steensgaard-Madsen, Kari Systa, Claus Unger, Dan Vodislav, Josef Voss.

1 INTRODUCTION

The workshop discussed what to be understood by a formal design method, and what is required to make the formal methods useful and used in practice, i.e., in design of industrial scale interactive systems. It is proposed to arrange a rather detailed comparison of several methods on a selected test case.

Many different aspects of formal methods could be considered under the heading of this workshop:

- What are the different types of formal methods or models especially suited for designing interactive systems?
- How to decide the degree of detail (or the level of abstraction), where a formal method is useful?
- What kinds of properties are expressable in each type of model?
- Does the choice of a formal model restrict the designer? In what ways?
- How can heuristics be combined with formalism?
- Does use of a formal model include user modeling?

L. J. Bass et al. (eds.), *Engineering for Human-Computer Interaction*

- What is the prime purposes of using formal methods?
  - Safe and/or efficient design and development?
  - Reproducable and documentable design and development?
  - Reasoning about a specification and verification of properties at an early stage of development?
  - Education of developers?
- Where should formal methods be used?
  - Making task models?
  - Specifying usability requirements?
  - Specifying dialogue control, especially temporal aspects?
  - Specifying graphical user interfaces?
- Can formal methods be usesd to real, industrial size systems or only to toy systems?

These questions point in many different directions but were used by the group for initial brainstorming, after which we selected and discussed a few of the questions.

2 WHAT IS A FORMAL METHOD?

The group reached reasonable consensus on the following description:
A *formal design method* is a development methodology based on a model that allows reasoning about interaction properties. The method

- does not include a 'full' user model, but it includes as many aspects of human behaviour as possible, i.e., such aspects that allow formalization;
- has a specification language with maximal power to capture interaction properties and to predict system interface behaviour; the language must have a precisely (mathematically) defined semantics;
- allows the construction of tools assisting development and tools assisting verification of usability properties. (Verification here means any formal or informal reasoning that can convince the designer and others that a system possesses some wanted properties.)
- can be used on large (industrial scale) systems.

Some keywords in this description are *reasoning*, *predicting*, and *verification* because a formalism is of no use unless it helps the designer to recognize and reason about structures

in the planned system. The description also underlines that the formal method should be especially oriented towards capturing the interaction aspects of a system.

3 MUST FORMAL METHODS BE USED BEFORE IMPLEMENTATION?

A discussion about the necessity of using a formal method when designing an interactive system led to the question: Is it a must always to use formal methods before implementation is begun?

The supporters of formal methods found the following reasons for answering yes to this question:

- It is the only way to make task analysis precise.
- When a design is stepwise refined, we must be able to prove that requirements still hold for the refined versions.
- With a formal specification you can reason about its properties.
- Some sort of formalism is necessary in order to compare and discuss different solutions to the same problem.
- Formal models and methods is necessary mental background for developers, and too much software is developed without any model.

The opponents came up with a number of arguments for not using formal methods:

- A formal method may lead to overconstrain in the design.
- Is the use of a formal design method compatible with iterative development?
- Sometimes you don't know what you want the system to do beforehand.
- It is hard to validate a formal specification with users, and the user feedback is critical when developing an interactive system.
- Are there any large systems where the development really started with formal methods?
- Practitioners hate formal methods, and the subject is not taught properly at our universities (at least in US).

Three strong arguments for, why formal methods are not (or very little) used for interactive systems in practice today, are: (i) formal methods are not taught properly at

universities; (ii) we don't have enough real success case stories; and (iii) existing formal methods are not supported by effective tools.

4 MUST FORMAL METHODS BE 'ADAPTED' TOWARDS PROGRAMMERS OF TODAY OR VICE VERSA?

One way to make formal methods more useful in the real world of today's programming could be that the inventors of formal methods (the scientists) adapted their methods better, e.g., by 'hiding all the mathematics'. But another way could be that programmers and designers must learn to appreciate the value and the benefits of using formal methods.

The group found a number of ways in which the university people must adapt their methods and improve their 'sales arguments':

- The formal notations must be re-written or simplified to be more easily understandable for non-scientists.
- The scientists must select more carefully those areas where formal methods can be really useful. Often a formal method should be applied only to selected parts of a system design.
- The scientists must present/teach the methods and their application better than yesterday and today.
- The scientists must demonstrate that formal methods can be realistically applied to industrial scale problems and not only toy problems.

But programmers and designers must 'move towards formalism' in the following ways:

- The practitioners must be taught the necessary concepts and vocabulary, so that they can better understand the formal methods.
- The practitioners and their managers must realize that for many systems (especially safety-critical real-time systems) correctness is more important than short term development efficiency.

The main difficulties in getting 'the gospel of formal methods' accepted by practitioners seem to be that (i) we don't have *one* formal theory commonly agreed upon among scientists, and (ii) we have not yet demonstrated the economic benefit of using formal methods.

5 COMPARISON OF FORMAL METHODS: A NEW WORKSHOP?

We discussed advantages and disadvantages of some formal methods presented by group members with the purpose of comparing good and bad aspects of the methods. It gave rise to interesting exchange of opinions, but in order to get a more thorough and concrete comparison of different formal methods we suggest:

> Arrange a workshop where several formal specifications (prepared as homework) of a specific system are presented and compared in detail.

More concrete we suggest:

1. Invite people to a workshop under CHI'95 (or some other conference) on the topic: Case study comparison of formal methods for interactive system design.
2. A number of (or all?) the participants must do homework by making a design based on their formal method of a specific problem.
3. The specific problem is: Specify an ideal NetScape-like browser and a server for WWW. Answer a number of questions for your specification:
   (a) Can you verify or disprove a number of usability properties for your specification? Here follows a list of more detailed questions about History, Browsability, Insistence, Non-preemptiveness, Exit/Stop at any time, Informative Feedback, Concurrency, ...
   (b) Is your specification executable? How difficult will it be to implement it?
   (c) Is your specification understandable/explainable to others? How is it structured?
   (d) Is your specification complete? Does it specify the entire system as envisaged?
   (e) How difficult will it be to modify the specification to include a specific new feature?
4. The specifications and the results of the workshop are published afterwards.
5. Fabio Paterno, Philippe Palanque and Gregory Abowd prepare an announcement of the proposed workshop and make preliminary arrangements with the CHI'96 program committee or some other conference program committee.