17

A User Model Supporting Communication in High-Level Design Systems

Anja Rockenberg, Susanne Heymann, Ursula Westerholz
Institute for System Design Technology
GMD-SET
Schloß Birlinghoven
D-53754 Sankt Augustin
e-mail: {rockenberg | heymann | westerholz}@gmd.de

1 Abstract

Today's high-level synthesis systems provide means to solve complex design problems but leave the user alone with an increasing number of difficult design decisions. We introduce a user model derived from requirements resulting from discussions with users and examination of existing systems. An implementation of this user model based on an existing high-level synthesis environment is sketched.

2 Introduction

Current frameworks and design systems administrate team work by providing intricate access and concurrency control mechanisms. In these models, the user is seen as an isolated person communicating with others exclusively through the use of common design objects. This view does prevent data collisions but it does not support human interaction in a working environment. Experience shows that good communication in a design team, the sharing of knowledge and results, increases work efficiency but on the other hand, people tend to work out a problem for themselves instead of asking a colleague who might already have found a solution. It follows that a design environment which takes user communication into account should provide means to share and distribute user experience within the system. This becomes increasingly important in high-level synthesis systems that require a lot of user interaction and expert knowledge in optimization steps.

We propose a user model introducing the concept of *virtual roles* to satisfy the need for communication between users in design frameworks. Our model is based on a small design team of up to eight people. Many state-of-the-art synthesis environments are tailored for single-user utilization, assuming a user that is well versed in all aspects of design. This is unrealistic – it is more likely that several people will pool their knowledge to achieve a design goal. On the other

F. J. Rammig et al. (eds.), *Electronic Design Automation Frameworks*

hand, due to the tight interaction between design steps, this group of people will not be very large.

This paper is structured as follows: We take a look at related work in the next section. This is followed by a definition of basic concepts for our work. Section 5 describes our sample system and the scenario from which we derive the requirements for our user model in section 6. A first implementation is presented in section 7. Our conclusions and an outlook to further work are presented in section 8.

3 Related Work

Cooperation of users in a design system is commonly seen as an access problem. First of all, users hold different levels of responsibility in their departments. These responsibilities vary according to the tasks a person is participating in and are reflected in access rights to task specific data. Secondly, while working together on a design, users are bound to work with tools that handle the same design data. Several approaches exist that address this kind of issue ([3], [1]). Our approach supports cooperation not just in respect of data management but on a more global scale. Conventional frameworks only consider conflicts as they appear; we introduce communication mechanisms that allow users to recognize impending conflicts and act to prevent them.

If work in a small design team is to be partitioned effectively, this partitioning must be comprehensible to every individual designer. The design system must allow the designer to monitor progress of the complete task and structure his own work according to his needs. Software engineering environments like ADDD [6] support services like this.

Design management also deals with assistance in design flow issues and project management. The JESSI Common Framework provides the means to define projects and design flows ([7], [10]) that form a static work environment. Other tools merely record the design process to allow later replay or assistance in tool selection [2]. Predefined and prerecorded flows are a valuable assistance to designers because they provide a process template that precludes erroneous design steps. However, even in a static design flow there may be different paths to one design goal. Users – especially those not acquainted with the design system – will want to know which of the paths yields the best result. Other users' previous experience can help in this case if it is made available within the whole of a design task. We propose a new service that accomplishes this.

4 Tasks and Roles

A *task* is a set of design and management activities that lead to the accomplishment of a design goal. Within a task, activities may again be merged into *subtasks*. Since these may be divided to further structure the design process, a subtask hierarchy is finally achieved.

Users are assigned to subtasks. They may organize work within their subtask according to their own needs. Subtasks may be delegated from one user to another under certain circumstances.

Users working together on a task form a *team*. Every member of a team has a set of *roles* that define his rights and responsibilities within the task.

There are several possibilities for modelling roles. We realize roles as relationships between users and tasks.

5 Sample System Scenario

Our sample system consists of a high-level synthesis design environment, incorporating several synthesis tools and various optimization algorithms. The tasks of high-level synthesis systems are presented in detail in [3], [8], and [12]. We consider a small group of users that works in a tight team to design an embedded system.

The high-level synthesis system CASTLE (Codesign And Synthesis TooL Environment) [11] serves as a basis for this scenario. A very condensed form of the design flow is depicted in Figure 1.

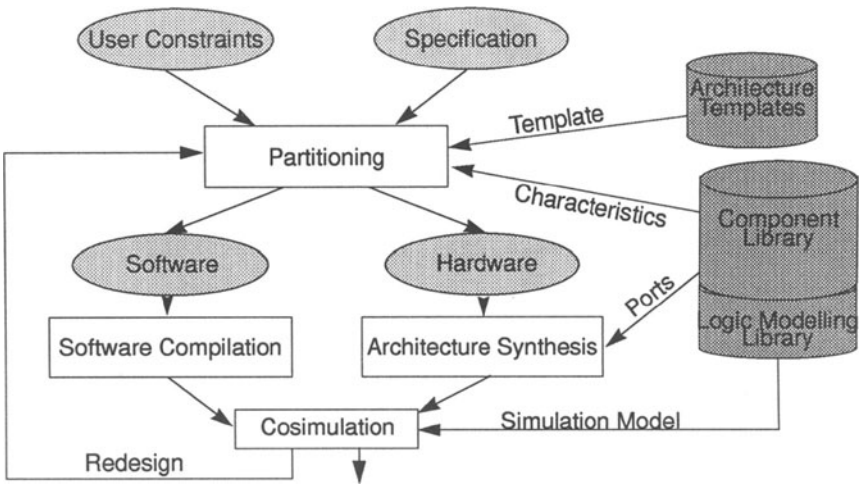

Figure 1. Prototype Scenario

We consider a team of two users working on the development of an MPEG video compression system. The functional description is available in C/C++. The conversion of this description to the CASTLE Synthesis Intermediate Representation format SIR is the first step of this task. This description is submitted to a profiling phase in the SIR/CASTLE Analysis Environment where information about operation count, instruction mix, most demanding functions, etc. is obtained. Based on these results, a design space exploration is performed and the principal system architecture is selected. In addition to this architecture template, a component library provides characteristics of processing components to fill the design with existing hardware structures.

This leads to the partitioning step where software and hardware parts are separated. During this process, the user should also consider information previously obtained by other users working on the same topic. Software is then directly compiled on the selected hardware architecture.

Additional application-specific hardware (e. g. ASICs, FPGAs) is synthesized and the joint result of these processes is subsequently verified by a cosimulation step using simulation models from the component library. The result of the cosimulation will normally lead to a re-partitioning of the system.

Processor synthesis and cosimulation are performed by commercial tools. Since this process is extremely time consuming, one wants to keep the probability of a redesign as small as possible. One means to achieve this in our small team is to have one designer perform compilation and synthesis steps while the other keeps trying to achieve better results in the analysis and partitioning step. If results that better match the design goal are found, they have to be communicated directly to the other team member, so that he may react accordingly.

Results are provided as a netlist of processing and storing components and their interface structure as well as the corresponding software components in C or Assembler.

A detailed design process using CASTLE is described in [14] and [15].

6 Requirements for Roles in a High-Level Synthesis Environment

In our sample environment, we consider a small team of up to eight users. Design steps in a synthesis system are closely interrelated: For example, any change in the partitioning decision immediately affects the software and hardware parts of the system to be built. In order to optimize the final design according to user constraints like performance or power consumption, several recursions of parts of or even the whole design process have to be performed. This process is highly interactive. Different optimization algorithms exist that may be chosen, and their results again depend on selecting the proper input parameters.

This close connection between design steps is reflected in the need for intensive cooperation between users because actions performed by one user will directly affect the results obtained by others working in the same task. Apart from the design-related work described above, there is the need for a coordinating agent. For our small team, we therefore derive the following roles:

- *Task Manager*: He supervises and coordinates activities within the design task. This includes the definition of subtasks and their delegation to designers as well as monitoring the progress of the task and its' results.
- *Designer*: Within his subtask, a designer must have as much flexibility as possible. He must be able to configure his environment, up to choosing his preferred tools or algorithms for a design step which offers alternatives in that respect. In addition, he may create his own subtask hierarchy to organise his work, delegate subtasks to other designers unless the task manager vetoes this action, and keep track of the progress of the whole task.

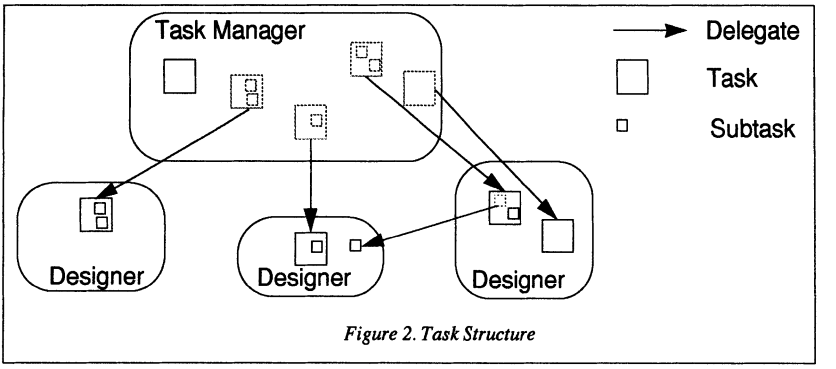

Figure 2. Task Structure

6.1 Virtual Roles

In addition to the roles fulfilled by real persons, we define two *virtual roles* that may be taken on by users as well as active system management objects. Virtual roles are initially delegated to the system but may be assigned to expert users who take over responsibility once they are chosen.

Virtual roles serve as key components in our approach and therefore require our special attention. They provide a simple and elegant means to model cooperation and communication as well as guidance in a design system like that described above where activities and their results are closely linked.

6.1.1 Moderator

The moderator is responsible for communication within a task. There is one moderator per task, responsible for recognizing communication needs and providing communication channels between users. A user must be able to tell the moderator that he does not want to be contacted directly at times. In that case, indirect mechanisms like mail may be used.

More specific, the communication aspects we envision with the moderator are organisational questions like task schedule, partitioning of a task, and the delegation of subtasks as well as conceptual questions like reactions to data collisions. In case of data collisions, new functionality for the solution of conflicts is provided. The moderator detects the need for communication in all of these cases and establishes the appropriate channels - either point-to-point or broadcast.

6.1.2 Adviser

The adviser provides an extended help service for users of the design system, ultimately combining help on framework services with application and design specific help. A hypertext-based, context-sensitive help server provides framework specific help. The integration of tool manual pages and help pages on design steps completes the system's part in this service. On request, default tool parameters and suggestions for design steps are provided, based on the evaluation of trace files. In addition, human users may register to the adviser as experts for information relevant to the task and system.

Contrary to common help servers, this mechanism provides the opportunity to reach information that is not written down formally but nevertheless may be important for accomplishing a design task. The adviser in this case points the user to the person who may be able to answer his question. This pointer may be directed to a specific help file maintained by the expert, giving hints to the user and protecting the expert from too many requests. If this help is not enough and direct communication is requested, the adviser signals the moderator to establish a communication channel to the expert who is contacted directly if he has not registered with the moderator as 'not available'.

7 Implementation of the User Model

Since the current CASTLE environment consists of stand-alone tools, our first implementation of the adviser and moderator roles is not based on an existing design framework. Instead, we use NCSA Mosaic [9] to display the adviser documents. The combination of Mosaic and HTML (HyperText Mark-up Language) easily allows to create a stand-alone adviser system. Forms are used for registering to the system as expert. Experts may insert their own documents into the adviser, realizing 'Frequently Asked Questions' pages that may be consulted before a human is actually contacted.

Further sources can be provided if the design system runs on internet-connected machines. In that case, access to the World Wide Web information services can be established.

The moderator role is realized as a suite of PERL [13] scripts that monitor adviser communication requests on the one hand and design file access on the other. Communication lines are established if the need is detected – either a button click in an adviser window or attempted access to a locked file.

8 Conclusions and Further Work

Acceptance of a system like that described above is difficult to measure! One major point of doubt was that users might refuse to register as experts to the adviser because they are afraid of constant interruptions of their own work as questions from others are flooding in. It turned out that in the small teams we have investigated here the psychological barrier to registering is quite low because people are well acquainted and need to work closely, anyway.

Novice users, e. g. students, needed less time to familiarize themselves with the CASTLE tool suite. Since the advisor/moderator duality is open and flexible to incorporate further domains, it can be extended to embrace advice on e. g. software engineering topics.

First results and feedback from users prompt us to extend the system. Next steps will be the integration of a WAIS database to better coordinate search for specific topics. In addition, we plan to integrate our service into an existing EDA framework.

It has been pointed out to us that the adviser service might be extended to encompass larger groups of users and a greater variety of topics. While this is technically possible we believe that there are limits to the amount of support that can be achieved. As the group of users accessing

one adviser increases, so does the overhead for experts. It remains to be investigated how big the supported group of users may grow in order for the adviser to work sensibly.

9 References

[1] K. O. ten Bosch, P. Bingley, P. van der Wolf: "Design Flow Management in the NELSIS CAD Framework", *Proc. of the 28th ACM/IEEE Design Automation Conference*, 1991.

[2] A. Casotto, A. Sangiovanni-Vincentelli: "Automated design management using traces", *IEEE Transactions on CAD*, August 1993.

[3] D. Gajski, N. Dutt, A. Wu, S. Lin: "High-level Synthesis - Introduction to Chip and System Design", *Kluwer Academic*, 1992.

[4] A. van der Hoeven, O. ten Bosch, R. van Leuken, P. van der Wolf: "A Flexible Access Control Mechanism for CAD Frameworks", *Proc. European Design Automation Conference*, 1994.

[5] H.-U. Kobialka, C. Meyke: "Configurations are Versions, too", *4th International Workshop on Software Configuration Management*, 1993.

[6] H.-U. Kobialka, C. Meyke: "Views on an Object Oriented Software Engineering Environment", *Proc. of the 6th International Workshop on CASE*, 1993.

[7] D. C. Liebisch, A. Jain: "JESSI Common Framework Design Management - The Means to Configuration and Execution of the Design Process", *Proc. 1st European Design Automation Conference*, 1992.

[8] P. Marwedel, W. Rosenstiel: "Synthese von Register-Transfer-Strukturen aus Verhaltensbeschreibungen", *Informatikspektrum 15: 5-22*, 1992.

[9] The National Center for Supercomputing Applications: "About NCSA Mosaic for the X Window System ", *http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html*, 1994.

[10] W. Schlegel (ed.): Design Management: JCF R4.0 Detailed Functional Specification - Draft Version, *JCF Document No. JCF/SNI/057-03/15-Jul-94*, 1994.

[11] M. Theißinger, P. Stravers, H. Veit: "Castle: An Interactive Environment for HW-SW Co-Design", *Proc. of the 3rd International Workshop on Hardware/Software Codesign*, 1994.

[12] R. A. Walker, R. Camposano (ed.): "A Survey of High-Level Synthesis Systems", *Kluwer*, 1991.

[13] L. Wall, R. L. Schwartz: "Programming perl - The Camel Book", *O'Reilly & Associates*, 1991.

[14] J. Wilberg, R. Camposano, W. Rosenstiel: "Design Flow for Hardware/Software Cosynthesis of a Video Compression System", *Proc. of the 3rd International Workshop on Hardware/Software Codesign*, 1994.

[15] J. Wilberg, R. Camposano, Ursula Westerholz, Uwe Steinhausen: "Design of an Embedded Video Compression System - A Quantitative Approach", *Proc. of the International Conference on Computer Design*, 1994.