1996

Guest Editor's Introduction

Luqi

Kluwer

Luqi, "Guest Editor's Introduction", Journal of Systems Integration. (Special issue on Computer-Aided Prototyping), 1996, Vol. 6, No. 1-2, pp. 15-17.
https://hdl.handle.net/10945/65702

Journal of Systems Integration, 6, 15–17 (1996)

Guest Editor's Introduction

The goal of research on computer-aided prototyping is to provide tools for developing reliable software systematically with high productivity. This area has strong connections with many other areas of software engineering, because it must face many of the same problems as other software development methods. For this reason, our goal of building a Computer-Aided Prototyping System (CAPS) required us to study software in its context of system engineering, and to consider software engineering issues ranging from requirements and domain modeling to software synthesis, analysis, reuse, and evolution, and also required us to integrate diverse systems addressing those issues.

Often, difficulties with requirements for new systems only appear when clients actually use the system. Rather than throwing away an initial full-scale implementation, it is better to develop requirements incrementally through inexpensive prototypes. This reduces the cost and increases the value of the envisioned system to the people it serves. Computer aid for rapidly and inexpensively constructing and modifying prototypes makes this feasible.

Evolution, which is the growth and change of systems, plays a key role in both computer-aided prototyping and in system engineering. Computers can support evolution by retrieving suitable reusable components from a software base, by merging independent changes to software components, by tracing dependencies to determine which parts of a software system are afffected by a requirements change, by using constraints to prevent errors by completing partial designs, and by generating programs, e.g., executable schedules for meeting real-time constraints or for gluing components together. Furthermore, methods and tools developed to support prototyping can often be applied to the ongoing evolution of mature software systems.

The CAPS project has focused on computer support for designing, building and modifying large real-time systems, because requirements for such systems can be especially troublesome. Begun more than a decade ago, this effort has addressed all the areas mentioned above, as well as prototyping language design, engineering databases, and project coordination based on computer-aided design and manufacturing.

The Papers

The papers in this special issue illustrate the diversity of the connections between prototyping and other areas of software engineering and some of the research progress in these subareas of system engineering:

1. *A Model-Based Computer-Aided Prototyping System*, by Li and Ketabchi, considers how domain models can be used to rapidly construct and exercise prototypes in an

object oriented framework. The MB-CAPS system has a graphical interface designed for problem domain experts who may not be programming experts.

2. *Real-Time Scheduling for Software Prototyping*, by Luqi and Shing, explains how hard real-time constraints can be realized for prototyping systems. This paper presents some practical scheduling algorithms for the single and multiple processor cases, and assesses their effectiveness.

3. *Translating EQL Programs for Bounded-Time Execution*, by Mok and Wangs, shows how to realize rule-based programs that must complete execution within fixed bounds on computation time. The paper gives both a method for translating a rule based program into programing code and a method for determining a bound on the execution time of the resulting target code.

4. *Software Component Search*, by Goguen *et al.*, suggests improved methods for finding reusable software components for a given design. The user's query is a syntax declaration and a set of test cases. Incrementally ranked multi-level filtering finds a small set of most promising components. Users do not need specialized skills in module semantics.

5. *Software Merge: Combining Changes to Decompositions*, by Berzins and Dampier, addresses a subproblem of software evolution, providing a method for combining changes to a software design represented by a hierarchy of annotated dataflow diagrams. This method addresses changes to the structure of a design, as well as changes to the specified system behavior.

Progress to Date

The prototyping language of the CAPS system, called PSDL, has been designed, formally specified, and partially implemented. Its semantics covers single and multiple processor implementations in a possibly distributed environment. Implementation techniques include program generation, automated scheduling, and reuse. One reuse method takes the design used for constructing a prototype as the basis for a query. Software evolution models have helped in developing automated methods for configuration management, team coordination, and semantics based merging of changes to prototypes. Summaries and references to papers describing these results are accessible over the world wide web at

```
http://www.cs.nps.navy.mil/research/caps
```

and

```
http://caps.airmics.gatech.edu/caps.html
```

Release 1 of the CAPS system is available free of charge from DISA (703-681-2364 or email to dsrscao@ssed1.ims.disa.mil), and from the Ada Joint Program Office (800-ADA-IC11).

CAPS has been used to develop a variety of prototypes, including a robot controller, a fish farm controller, a simple autopilot, a controller for a hyperthermia cancer therapy system, a secure communications link, a generic C3I system, a land to air missile, and a cruise missile guidance system.

Ongoing and Future Research

The CAPS project is currently working to improve support for software evolution, software reuse, program generation, real-time scheduling, and configuration management. One interesting topic for future research is the transition from prototypes to final products. Progress here could remove barriers between prototyping and product development, and success could revolutionize the way the software industry does business, by introducing much higher levels of automation. Research issues here include the following: (1) optimization of prototype implementations, (2) transforming prototypes to run on hardware and operating systems different from those of the prototyping environment, (3) switching from simulated external systems to actual ones by generating appropriate concrete interface programs, (4) introducing data persistence and scaling up to large volumes of data, (5) realizing fault tolerance, (6) realizing security constraints, (7) certifying the integrity of designs, and (8) checking that implementations realize designs.

The advances in computer-aided prototyping, partially reported in this issue, have been made possible by the contributions of a few hundred researchers and graduate students, and reflect Raymond Yeh's vision of twelve years ago. It is due to the trust, encouragement and strong support from Jack Schwartz, Dan Berry, KC Tai, CV Ramamoorthy, Sartaj Sahni, JB Rosen, David Hislop and many others, from the time when the CAPS approach to software development was a little "ugly duckling". I am grateful to all these people.

Luqi
Computer Science Department,
Naval Postgraduate School,
Monterey, CA 93943