



Calhoun: The NPS Institutional Archive
DSpace Repository

Reports and Technical Reports

All Technical Reports Collection

1990

Rapid Prototyping for Software Evolution

Luqi, Tanik, M.; Yin, W.

Naval Postgraduate School

Luqi, M. Tanik, and W. Yin, "Rapid Prototyping for Software Evolution", Technical Report NPS 52-90- 009, Computer Science Department, Naval Postgraduate School, 1990.
<https://hdl.handle.net/10945/65202>

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>

T51

NPS52-90-009

NAVAL POSTGRADUATE SCHOOL

Monterey, California



RAPID PROTOTYPING FOR SOFTWARE EVOLUTION

W.P. YIN, LUQI and M.M. TANIK

August 1989

Approved for public release; distribution is unlimited.

Prepared For:

National Science Foundation
Washington, DC 20550

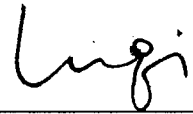
NAVAL POSTGRADUATE SCHOOL
Monterey, California

Rear Admiral R. W. West, Jr.
Superintendent

Harrison Shull
Provost

This report was prepared in conjunction with research funded by the National Science Foundation.

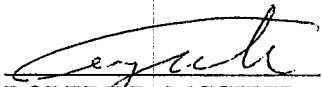
Reproduction of all or part of this report is authorized.

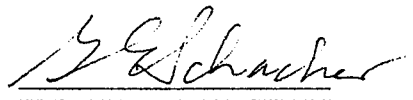


LUIGI
Assistant Professor
of Computer Science

Reviewed by:

Released by:

for 
ROBERT B. MCGHEE
Chairman
Department of Computer Science


KNEALE T. MARSHALL
Dean of Information
and Policy Science

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPS52-90-009		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Computer Science Dept. Naval Postgraduate School	6b. OFFICE SYMBOL (if applicable) 52Lq	7a. NAME OF MONITORING ORGANIZATION National Science Foundation & ONR	
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943		7b. ADDRESS (City, State, and ZIP Code) Washington, DC 20550	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION National Science Foundation	8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER NSF CCR-8710737	
8c. ADDRESS (City, State, and ZIP Code) Washington, DC 20550		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO.	PROJECT NO.
		TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) RAPID PROTOTYPING FOR SOFTWARE EVOLUTION (U)			
12. PERSONAL AUTHOR(S) W.P. Yin, Luqi, and M.M. Tanik			
13a. TYPE OF REPORT Progress	13b. TIME COVERED FROM Mar TO Dec 89	14. DATE OF REPORT (Year, Month, Day) August 1989	15. PAGE COUNT 13
16. SUPPLEMENTARY NOTATION			
17. COSATI CODES		18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP		
19. ABSTRACT (Continue on reverse if necessary and identify by block number) This paper describes the basic concepts of a mixed-level software process and rapid prototyping. Some practical methods, and software tools for using rapid prototyping to support software evolution are also presented. The research results show that prototyping can be used to stabilize the requirements for either an initial software development project or a proposed enhancement to an existing system.			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Luqi		22b. TELEPHONE (Include Area Code) (408) 646-2735	22c. OFFICE SYMBOL 52Lq

Rapid prototyping for Software Evolution

W. P. Yin[†], Luqi[†] and M. M. Tanik[†]
Department of Computer Science and Engineering[†]
Southern Methodist University
Dallas, TX 75275-0122

Department of Computer Science[‡]
The Naval Postgraduate School
Monterey, CA 93943

Abstract — This paper describes the basic concepts of a mixed-level software process and rapid prototyping. Some practical methods, and software tools for using rapid prototyping to support software evolution are also presented. The research results show that prototyping can be used to stabilize the requirements for either an initial software development project or a proposed enhancement to an existing system.

As society becomes increasingly depending on computers, the public demands for computer software becomes more complex than perhaps any other human construct [FPB87]. The facts of missed schedules, blown budgets, and flawed products have shown that engineering of software needs technology innovations to increase software productivity and quality. More and more computer scientists [RTY87, FPB87] believe that concentration on the front-end technologies, specification and design, are the first priority. The positive results in these fields have potential for breakthroughs in the engineering of software.

To achieve any significant gain in software productivity it has been proposed that an alternative software development and evolution is necessary [RTY87]. Figure 1 depicts such a paradigm. Software is a spectrum concept that covers all the information obtained during software developments and evolutions. The productivity of software engineering much depends on the software process model. Among a number of software process models, the waterfall model (Figure 2) has become the basis for most software acquisition standards [BWB88]. The experience has proved that the waterfall model has some fundamental difficulties, such as the inherent discontinuity among the

This research was supported in part by the National Science Foundation under grant number CCR-8710737 and Texas Instruments, Inc.

phases, the lack of good control in the front-end of the process, and costly product maintenance. In the alternative model, the concentration is shifted to the front-end, namely, requirement specification and design; and the mixed-level process conducts the prototyping, performance measurement, verification, code synthesis as well maintenance.

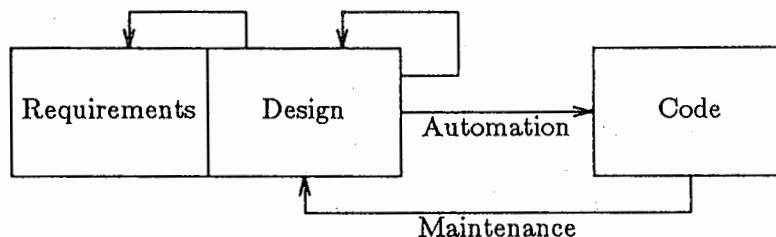


Figure 1. An Alternative Software Development Model

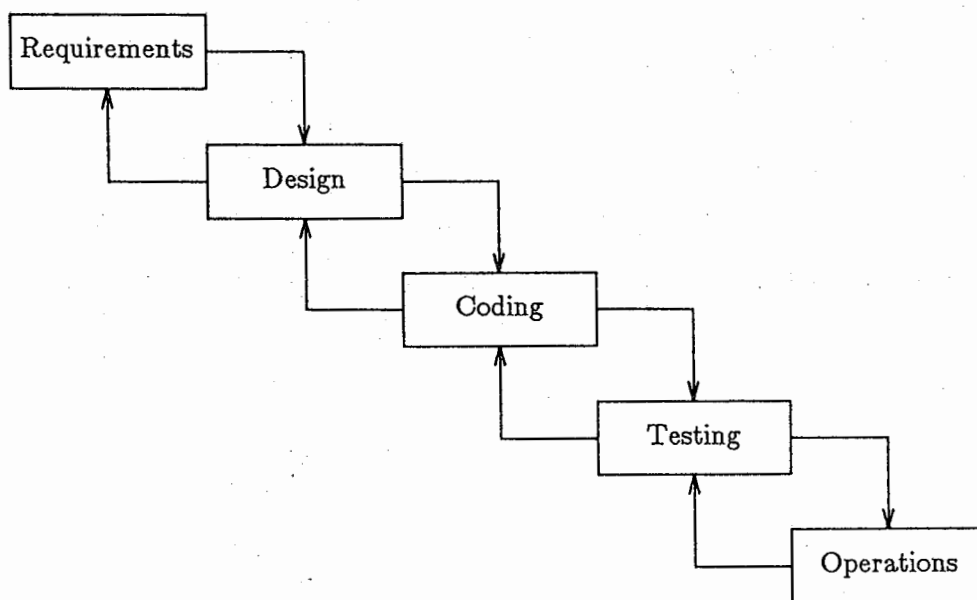


Figure 2. Standard Waterfall Life Cycle Model

In order to apply the alternative software process model, a number of technology innovations are required. In this paper, we represent a concrete mixed-level design process and a practical rapid prototyping system. The mixed-level design process (Figure 3) employs three design activities representing different levels of software development [T&Y89]. The activities are Construct Design, Exercise Design and Translate Design. These activities support the design evolution. Testing and

maintenance are performed directly on the design. Utilizing prototyping technique in the mixed-level design process obviously helps software evolution.

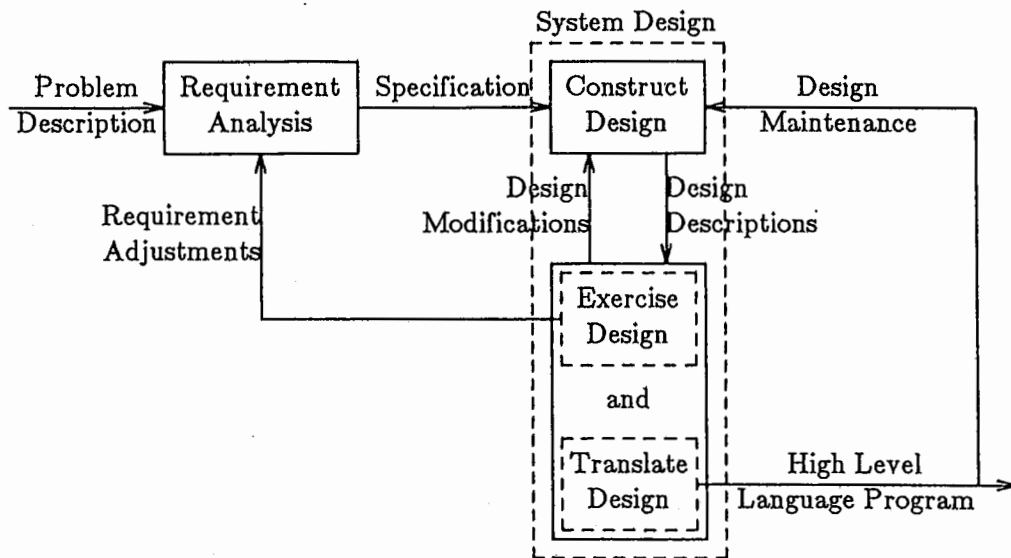


Figure 3. A Mixed-level Design Process

In our approach, a prototype is a concrete executable model of selected aspects of the proposed system, and rapid prototyping is the process of building and evaluating a series of prototypes rapidly. Figure 4 illustrates the iterative prototyping cycle. The user and the designer work together to define the requirements and specifications for the critical parts of the envisioned system. The designer then constructs a model or prototype of the system in a prototype description language at the specification level. The constructed prototype is a partial representation of the system, including only those attributes necessary for meeting the requirements, and is used as an aid in analysis and design rather than as production software. During demonstrations of the prototype, the user validates the prototype's actual behavior against its expected behavior. If the prototype fails to execute properly the user identifies problems and works with the designer to redefine the requirements. This process continues until the user determines that the prototype successfully captures the critical aspects of the envisioned system. Following this validation, the designer uses the validated requirements as a basis for the design of the production software.

A set of computer-aided software tools, the Computer-Aided prototyping System (CAPS) [L&K88], has been designed and integrated to support prototyping of complex software systems such

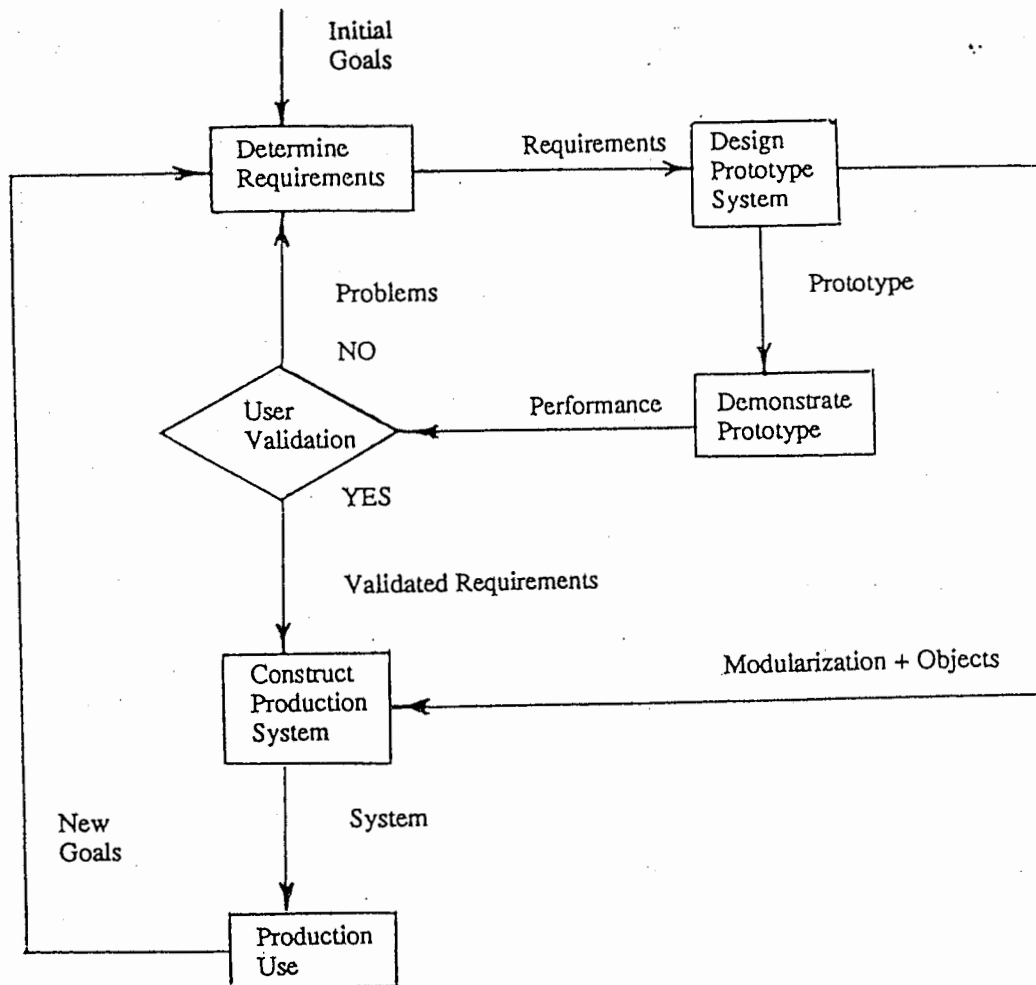


Figure 4. The prototyping Cycle

as control systems with hard real-time constraints. The requirements for such systems are especially difficult to determine and their feasibility is hard to establish without constructing an executable model of the envisioned system [LUQI89a].

The CAPS System contributes to software evolution by providing software tools for creating prototypes and adapting them to new requirements. If the prototyping process is carried out manually, the associated benefits are limited because it takes too much time and effort. CAPS can increase the leverage of the prototyping strategy by reducing the effort that must be spent by the designer in producing and adapting a prototype to perceived user needs.

The evolution of a prototype starts after one pass through the prototyping cycle shown in Figure 4: the analysts have determined the initial requirements by talking to the customers, constructed

an initial prototype, and demonstrated it to the customer, who finds some aspects of the prototype's behavior unacceptable and requests some modifications.

Initially, the facilities provided by CAPS are used to adapt the prototype to the new requirements. Modifications to the production software can be implemented by using CAPS to (1) add changes to prototype systems, (2) retrieve software components from the software base, (3) generate production code if needed, (4) assemble production systems via the prototyping cycle, and (5) manage the process via the prototyping database.

The main components of CAPS are a special prototyping language and a set of tools illustrated in Figure 5. The main subsystems of CAPS are the user interface, the software database system, and the execution support system. Those components are described in detail in [LUQI89b].

We have applied the rapid prototyping techniques to build a prototype in the philosophy of CAPS [WPY89]. The prototype is implemented by using knowledge-based techniques, representations and inference mechanism. It is build on the top of ART (The Automatic Reasoning Tools), and runs on TI Explorer under system software version 3.1. The structure of the prototype consists of the following components (Figure 6).

(1) The Uniform Design Representation:

The fundamental problem of a design process is finding the right representation of design information. A good design representation is the first step forward to a successful mixed-level design process. A uniform design representation is the only connection among the user interface, design construction, design exercise and design translation. With the uniform representation, each subsystem works independently without knowing the internal representation of all other subsystems and accessing data at all times in every internal status. Using a uniform representation is also reducing the number of transformation from one internal representation to another. In our prototype, the uniform design representation is a hybrid of object-oriented and constraint-based form.

(2) Static Analyser:

The analyser performs static dependency analyses of the software design, such as data dependency, control dependency, interface dependency, etc. These dependencies are useful at design validation and maintenance.

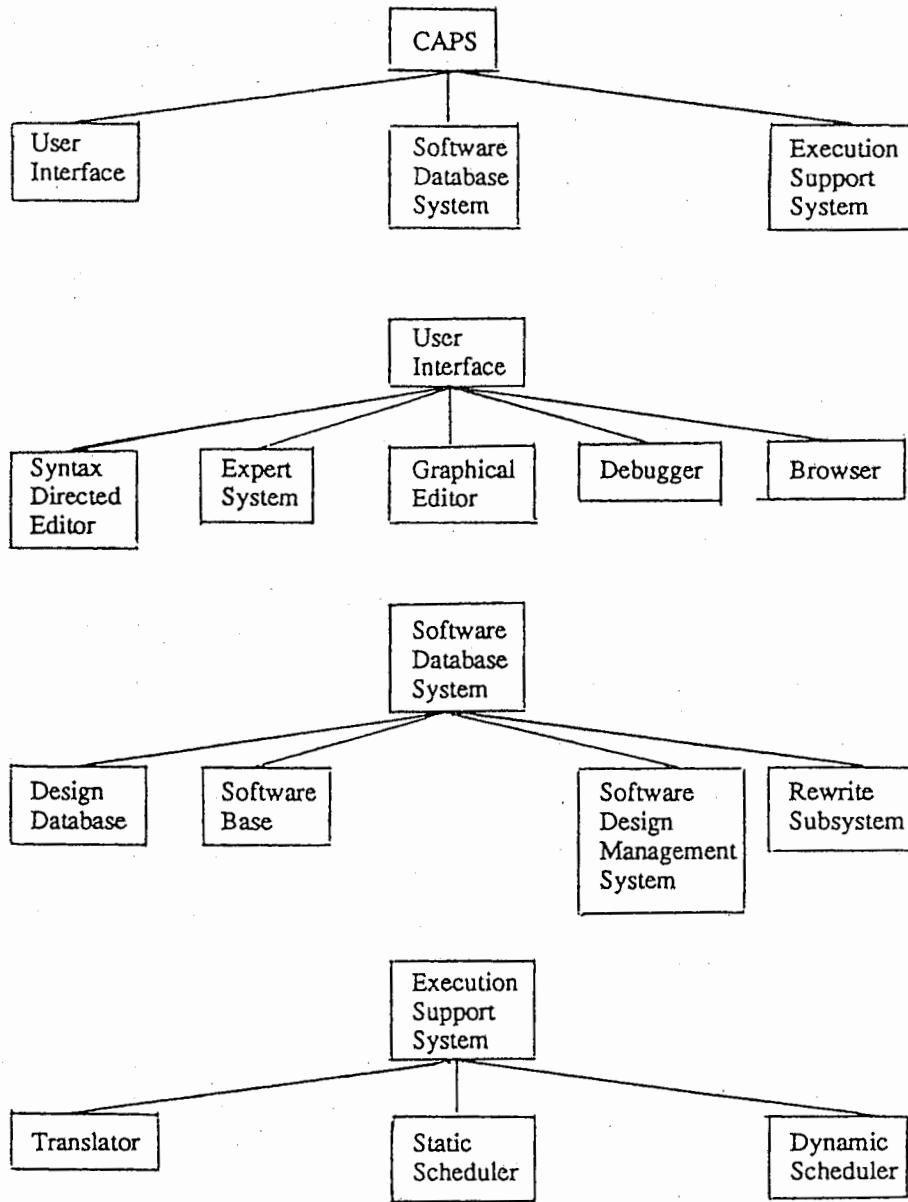


Figure 5. Main CAPS Tools

(3) Exerciser (Simulator/Interpreter):

The main function of the exerciser is to expose the behavior of the software system being designed and detect the design errors by dynamic analyses. Exposing the system behavior at design phase can give both customers and designers early feedback, reducing the cost of changes to the implementation.

(4) Automatic Program Generator:

After testing the design, the desired high level language program can be automatically generated from the design. The automation guarantees to reserve the designed functionalities. Direct modification on code is not necessary. The resulted program may also be executed by the exerciser to validate the intended design.

(5) Design Component Base:

The design component base stores and retrieves previous designed software components. Each design component in the base is described in the uniform design representation. The base management supports design maintenance and reusability directly.

(6) User Interface:

The main function of a user interface is to provide communications between the designer and design environment. The user interface offers the following facilities:

a. Design acquisition:

The design acquisition consists of graphics tools and editors. The editors cooperate with multiple external design representations and graphics tools, such as dataflow-oriented editor, state machine-oriented editor, language-oriented syntax-directed editor, etc. All those editors take different external design representations as inputs and convert them into the uniform design representation. The designer can choose an editor supporting the design methodology he is familiar with.

b. Testing Display:

The testing display shows the prototype execution or the interpretation of the design as well as the target program execution results. The display may be a time chart indicating the system state changes or the desired system behavior sequence like dialogue, input/output, reactions, etc.

c. Analysis Display:

The analysis display shows the results of the static analyses as the designer required.

d. Program Output:

The program generated based on the current status of the design can be retrieved by the designer and delivered to the customers.

According to the above discussion, the uniform design representation is the center of the design

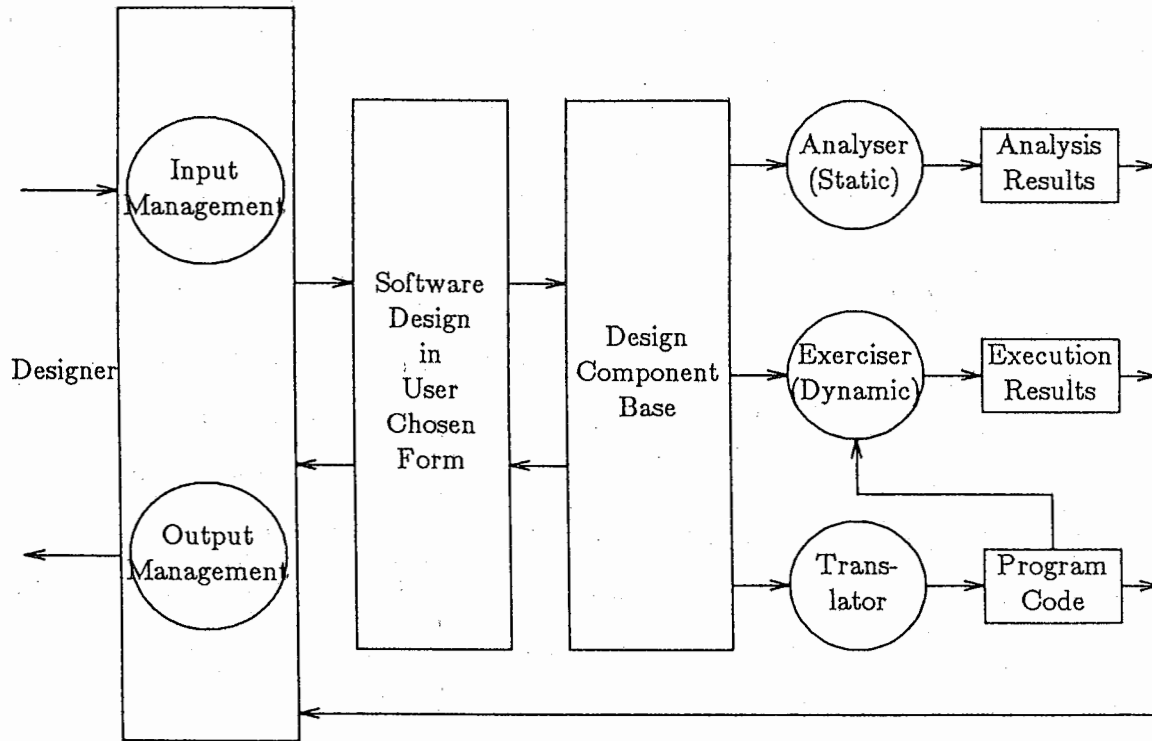


Figure 6. A General Structure of Design Environment

environment. All of the user preferred external information format will be converted to the uniform representation as the internal format, or vice versa, and all of subsystems operate on the internal representation, eliminating the redundant subsystems. Another advantage of using a uniform internal representation is to make an open architecture for the design environment in order to meet a variety of user interface requirements according to designer's preferences.

The results of our research work on a mixed-level software process represent an advancement in the state-of-art of software process model. The development of CAPS is a step toward many software technology innovations. The prototyping experience has demonstrated the great potential in software evolution. Although, much more need to be done, such as distributed information management, it is our belief that progresses would have to be made stepwise, at great effort, and persistence will eventually pay.

Acknowledgement — We have benefitted from Dr. Raymond T. Yeh's vision and direction in pursu-

ing this research.

- [BWB88] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, Vol.21, No.5, May, 1988.
- [FPB87] F. P. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *IEEE Computer*, Vol.20, No.4, April, 1987.
- [LUQI89a] Luqi, "Handling Timing Constraints in Rapid prototyping," *Proc. of HICSS-22*, Vol.2, 1989.
- [LUQI89b] Luqi, "Software Evolution Via Rapid prototyping," submitted to *IEEE Computer*, 1989.
- [L&K88] Luqi and M. Ketabchi, "A Computer Aided prototyping System," *IEEE Software*, Vol.5, No.2, March, 1988.
- [RTY87] R. T. Yeh, "Some Software Issues of Strategic Defense Systems," *Proc. of FJCC*, Oct. 1987.
- [T&Y89] M. M. Tanik and R. T. Yeh, "The Role of Rapid prototyping in Software Development," *Proc. of HICSS-22*, Vol.2, 1989.
- [WPY89] W. P. Yin, "A New Software Design Paradigm and Its Prototype," Ph.D. Dissertation, Department of Computer Science and Engineering, SMU, 1989.

DISTRIBUTION LIST

- | | | |
|-----|--|-----|
| (1) | Defense Technical Information Center
Cameron Station
Alexandria, Virginia 22304-6145 | 2 |
| (2) | Dudley Knox Library
Code 0142
Naval Postgraduate School
Monterey, CA 93943 | 2 |
| (3) | Center for Naval Analysis
4401 Ford Avenue
Alexandria, VA 22302-0268 | 1 |
| (4) | Director of Research Administration
Attn: Prof. Howard
Code 012
Naval Postgraduate School
Monterey, CA 93943 | 1 |
| (5) | Chairman, Code 52
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5100 | 1 |
| (6) | Chief of Naval Research
800 N. Quincy Street
Arlington, Virginia 22217 | 1 |
| (7) | National Science Foundation
Division of Computer and Computation Research
Attn. Tom Keenan
Washington, D.C. 20550 | 1 |
| (8) | Naval Postgraduate School
Code 52Lq
Computer Science Department
Monterey, CA 93943 | 100 |