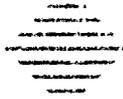


Technical Report  
CMU/SEI-92-TR-13  
ESC-92-TR-013

2



Carnegie Mellon University  
Software Engineering Institute

**AD-A258 466**



**A Classification and  
Bibliography of Software Prototyping**

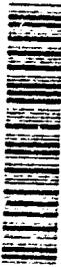
David P. Wood  
Kyo C. Kang

October 1992

**S** **DTIC**  
**ELECTE**  
**DEC 29 1992**  
**A** **D**

This document has been approved  
for public release and sale; its  
distribution is unlimited.

**BEST  
AVAILABLE COPY**

  
**92-32849**

**92 12 28 024**

1. The first part of the document is a list of names and addresses of the members of the committee. The names are listed in alphabetical order, and the addresses are given in full. The list includes the names of the members of the committee, the names of the members of the sub-committee, and the names of the members of the advisory committee. The addresses are given in full, including the street, city, and state.

2. The second part of the document is a list of the names and addresses of the members of the committee. The names are listed in alphabetical order, and the addresses are given in full. The list includes the names of the members of the committee, the names of the members of the sub-committee, and the names of the members of the advisory committee. The addresses are given in full, including the street, city, and state.

3. The third part of the document is a list of the names and addresses of the members of the committee. The names are listed in alphabetical order, and the addresses are given in full. The list includes the names of the members of the committee, the names of the members of the sub-committee, and the names of the members of the advisory committee. The addresses are given in full, including the street, city, and state.

Technical Report  
CMU/SEI-92-TR-13  
ESC-92-TR-013  
October 1992

# A Classification and Bibliography of Software Prototyping



David P. Wood  
Kyo C. Kang

DEPARTMENT OF COMPUTER SCIENCE

Requirements Engineering Project

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

Approved for public release.  
Distribution unlimited.

Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, Pennsylvania 15213

This technical report was prepared for the

SEI Joint Program Office  
ESC/AVS  
Hanscom AFB, MA 01731

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

#### Review and Approval

This report has been reviewed and is approved for publication.

FOR THE COMMANDER



Thomas R. Miller, Lt Col, USAF  
SEI Joint Program Office

The Software Engineering Institute is sponsored by the U.S. Department of Defense.  
This report was funded by the U.S. Department of Defense.

Copyright © 1992 by Carnegie Mellon University.

This document is available through the Defense Technical Information Center. DTIC provides access to and transfer of scientific and technical information for DoD personnel, DoD contractors and potential contractors, and other U.S. Government agency personnel and their contractors. To obtain a copy, please contact DTIC directly: Defense Technical Information Center, Attn: FDRA, Cameron Station, Alexandria, VA 22304-6145.

Copies of this document are also available through the National Technical Information Service. For information on ordering, please contact NTIS directly: National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

Copies of this document are also available from Research Access, Inc., 3400 Forbes Avenue, Suite 302, Pittsburgh, PA 15213.

Use of any trademarks in this report is not intended in any way to infringe on the rights of the trademark holder.

# Table of Contents

<b>Literature Identifier Directory</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Report Outline	1
1.2 A Suggested Starting Point	2
<b>2 A Prototyping Technology Framework</b>	<b>3</b>
2.1 Impetus for Early and Continuous Validation	3
2.2 Basis for Prototyping as a Software Engineering Paradigm	5
2.2.1 Classifying Prototyping Approaches	7
2.2.2 A Generic Prototyping Process	9
2.2.3 Common Prototyping Activities	10
2.2.4 Different Approaches for Different Needs	11
2.3 A Framework for Method and Tool Selection	12
<b>3 Summary and Directions</b>	<b>14</b>
<b>4 Taxonomy and Classification</b>	<b>15</b>
<b>5 Bibliography</b>	<b>23</b>
5.1 Bibliography Selection Criteria	23
5.2 Annotated References	24
<b>Index by Author</b>	<b>69</b>
<b>Index by Keyword</b>	<b>73</b>
<b>Index by Source</b>	<b>75</b>
<b>Index by Title</b>	<b>77</b>
<b>Index by Year</b>	<b>81</b>
<b>Additional References</b>	<b>85</b>



## List of Figures

<b>Figure 2-1</b>	The Importance of Early Requirements Validation	3
<b>Figure 2-2</b>	Types of Errors in Requirements	4
<b>Figure 2-3</b>	Evolving Mission Needs into Validated Requirements	6
<b>Figure 2-4</b>	Prototyping as a Continuous Process	7
<b>Figure 2-5</b>	Common Prototyping Activities	10
<b>Figure 2-6</b>	Examples of Prototyping Model Focus	11
<b>Figure 2-7</b>	Different Approaches for Different Needs	12
<b>Figure 2-8</b>	A Prototyping Technology Framework	13



## List of Tables

<b>Table 1</b>	Prototyping Approach Taxonomy	17
<b>Table 2</b>	Prototyping Literature Classifications	18



## Literature Identifier Directory

Andriole89	24	IEE89	42
Andriole90	24	Jain89	43
Andrulis90	24	Jones90	43
Bagrodia90	25	Koch88	43
Bailes87	25	Konrad	44
Bajwa89	25	Kordon90	44
Barbacci91	26	Kozubal90	45
Bartschi89	27	Kreutzer90a	45
Berardi89	27	Kreutzer90b	46
Biggie89	28	Krista89	46
Birch89	28	Lea90	46
Black91	29	Luckey90	47
Brooks87	29	Luqi86	47
CASEOut90	29	Luqi87	47
Chang90	30	Luqi89	47
Cohen90	30	Luqi90a	48
Cooling89	31	Luqi90b	48
Cooper90	31	Luqi91	48
Cordy91	32	Luqi92	49
Degl'Innocenti90	33	Madison89	49
Demeure89	33	Matsumoto	49
DeSoi89	32	McEney90	50
Diaz-Gonzalez91	34	McInroy89	50
Edmonson89	34	Milovanovic90	51
Ekambareshwar89	35	Minkowitz89	51
Espinosa90	35	Nugent88	52
Fisher87	35	O'Neil89	52
Ganti90	36	Ortner88	53
Gerber90	36	Overmyer90	53
Gimnich87	36	Powers89	54
Giordano91	37	Purtilo91	54
Gomaa81	37	Rizman90a	55
Gonzalez89	37	Rizman90b	55
Gregorio90	38	Royce89	56
Gutierrez89	38	Rzepka86	57
Harris87	39	SEKE90	57
Hartson91	39	Shirota89	58
Hawryszkiewicz87	40	Smith90	58
Heisler88	41	Smyrniotis90a	59
Hekmatpour90	41	Smyrniotis90b	59
Henskes87	41	Son88	60
Henskes90	42	Tamanaha90	60
Hughes89	42	Tenazas90	61

Thayer90	61
Trenouth91	62
Tsai89	62
Tsai90a	63
Tsai90b	63
Tucherman90	64
Turnheim89	64
Tyszberowicz89	65
Wallentinson89	65
Warkowski90	65
Wellner89	66
Whatmore91	66
Wing91	67
Zhao91	67
Zompi90	68

# A Classification and Bibliography of Software Prototyping

**Abstract:** Prototyping, the creation and enaction of models based on operational scenarios, has been advocated as a useful software engineering paradigm because it lends itself to intense interaction between customers, users, and developers, resulting in early validation of specifications and designs. An extensive and widespread interest in software prototyping in recent years has resulted in a daunting amount of literature and dozens of proposed methods and tools. As with any immature and growing technology, the expanding literature and approaches have resulted in correspondingly expansive and confusing terminology.

This report presents an overview of technology and literature relating to the creation and use of software system prototypes. In addition to an annotated bibliography of recent prototyping literature, a technology framework, taxonomy, and series of classifications are provided. The intent of this report is to provide a basic road map through the available literature and technology.

## 1 Introduction

The purpose of this report, compiled as a part of the work of the Software Engineering Modeling Project at the Software Engineering Institute (SEI), is threefold:

- A technology overview and framework are presented with the purpose of providing a basic road map through the available literature and technology.
- A taxonomy and classifications are presented to provide a means for structuring, managing, and selecting literature relating to software prototyping.
- An annotated bibliography is presented to provide an historical background on the field as well as necessary background for further work in the discipline both for projects both at the SEI and elsewhere.

### 1.1 Report Outline

This document contains the following major sections:

1. A prototyping technology overview, including a discussion of prototyping processes and activities and an abstract framework for method and tool selection (**Section 2, "A Prototyping Technology Framework"**)
2. A taxonomy of prototyping approaches and classifications of prototyping literature (**Section 4, "Taxonomy and Classification"**)
3. A set of full literature citations and annotations (**Section 5, "Bibliography"**)

4. A set of cross-reference indices to facilitate document search and identification, including:
  - An alphabetical cross reference by author's name ("**Index by Author**" on page 69)
  - An alphabetical cross reference by keyword ("**Index by Keyword**" on page 73)
  - An alphabetical ordering by publication source ("**Index by Source**" on page 75)
  - An alphabetical index listing by title and the page number where the full citation may be found ("**Index by Title**" on page 77)
  - A chronological ordering by year of publication ("**Index by Year**" on page 81)

## 1.2 A Suggested Starting Point

For those readers unfamiliar with software systems prototyping, the following references may provide a suitable starting point:

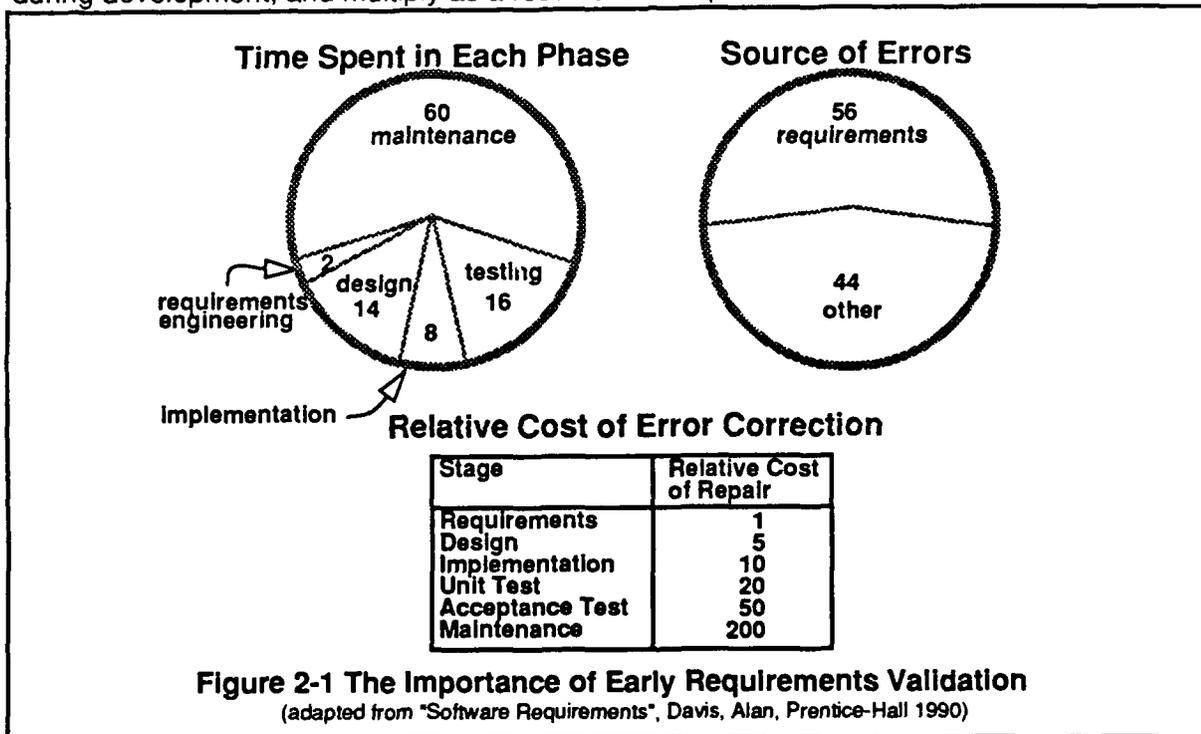
- General Overview
  - S.J. Andriole; *Modern Life Cycling: Some System Design Principles for the 90s* [Andriole90]
  - H. Gomaa and D.B.H. Scott; *Prototyping as a Tool in the Specification of User Requirements* [Gomaa81]
  - A. Hekmatpour and P. Chau; *AI Techniques and Object-oriented Technology for VLSI Design-space Representation, Optimization, and Management* [Hekmatpour90]
  - A.P. Sage and J.D. Palmer; *Software Systems Engineering* [Sage90]
- Acquisition and Standards
  - H. Black, D. Leciston, R. McGhee, and J. Zimmerlich; *Acquisition Models for the Capture and Management of Requirements for Battlefield Software Systems* [Black91]
- Process, Method, and Tool Support
  - F.P. Brooks, Jr.; *No Silver Bullet: Essence and Accidents of Software Engineering* [Brooks87]
  - CASE Outlook 90, No. 4; *Survey of Rapid Prototyping Tools* [CASEOut90]
  - R.H. Thayer and M. Dorfman; *System and Software Requirements Engineering* [Thayer90]

Additional suitable references have been noted in Table 2 under the **Roadmap** category.

## 2 A Prototyping Technology Framework

### 2.1 Impetus for Early and Continuous Validation

Although there have been substantial advancements in software engineering methods and tools during the past twenty years, requirements engineering still remains a key problem area in the development of complex, software-intensive systems. The report of the Defense Science Board Task Force on Military Software concluded that "the hardest part of the software task is the setting of the exact requirements" [DSB87]. One of the primary sources of continued difficulty is the lack of early requirements validation. Validation of requirements is problematic because requirements often are not well understood prior to development, change frequently during development, and multiply as a result of development.

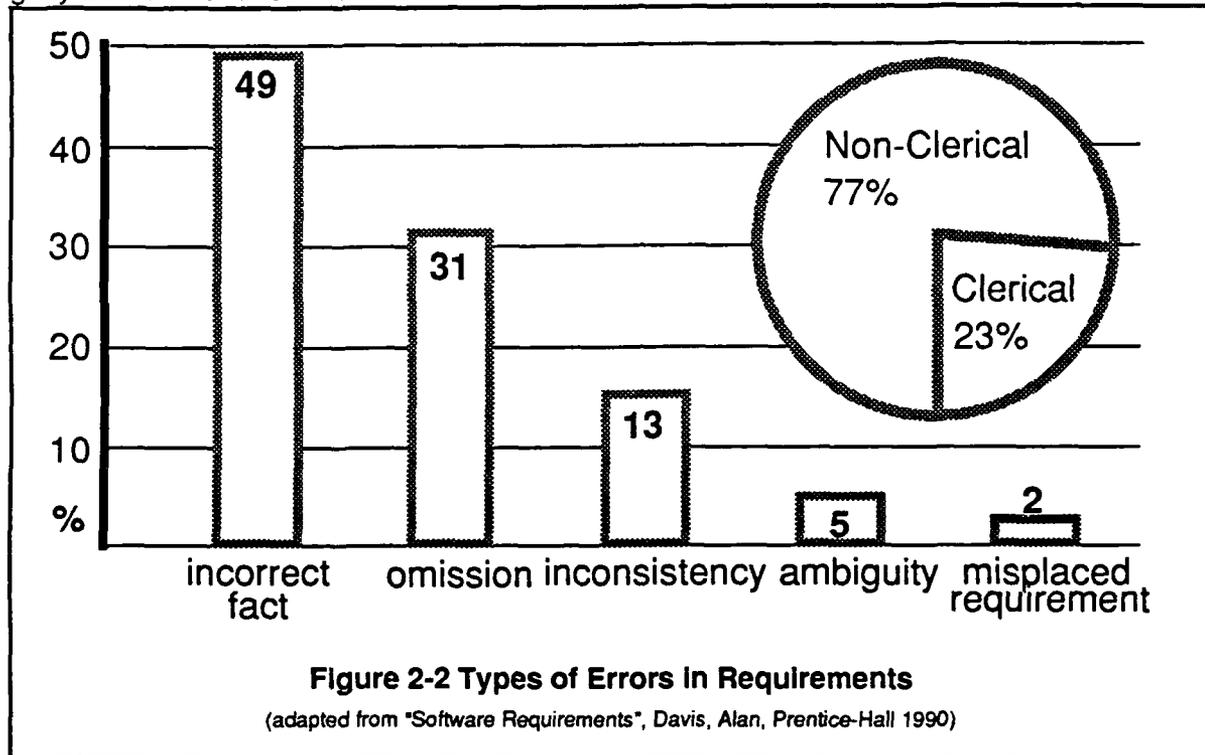


Statistical data has testified to the importance of early validation (Figure 2-1). Boehm reported that 54% of all errors ever detected in software projects studied at TRW were detected after the coding and unit testing stages and most of these errors (83%) were attributable to the requirements and design stages rather than the coding stage (17%) [Boehm75].<sup>1</sup> Also, DeMarco reported that 56% of all bugs detected could be traced to errors in requirements [Tavolato84]. Many requirements errors are passed undetected to the later phases of the life cycle, and correcting these errors during or after implementation has been found to be extremely costly [CONG90]. The DoD Software Technology Plan states that "early defect fixes are typically two

<sup>1</sup> While this data does not differentiate problems attributable to requirements from those attributable to design, it does serve to support the contention that errors discovered significantly later than their introduction are expensive to address.

orders of magnitude cheaper than late defect fixes, and the early requirements and design defects typically leave more serious operational consequences." [DOD91] Clearly, better techniques are needed for early validation.

Further, the types of errors most frequently made during requirements engineering have been found to be technical ones (Figure 2-2). Basili reported that 77% of all requirements errors found from the Navy A-7E aircraft's operational flight program were non-clerical errors, of which 49% were incorrect facts and 31% were omissions [Basili81]. Inconsistency and ambiguity account for about 18% of all non-clerical errors.



The *waterfall* life cycle model, which requires a complete requirements specification before development, also contributes to these problems [Royce70]. For large, complex systems, it is difficult to completely specify requirements in advance of and independent from design and implementation. The assumption that it is possible to create a complete specification prior to development has been a major cause of problems due to frequent changes to the specification during and after development.

## 2.2 Basis for Prototyping as a Software Engineering Paradigm

The validation of requirements early in the life cycle is one of the key issues in software development because failure to validate requirements can result in frequent and expensive changes in later life cycle phases. The DSB report concluded that:

"We believe that users cannot, with any amount of effort and wisdom, accurately describe the operational requirements for a substantial software system without testing by real operators in an operational environment, and iteration on the specification. The systems built today are just too complex for the mind of man to foresee all the ramifications purely by the exercise of the analytic imagination." [DSB87]

*Prototyping* has been discussed in the literature as an important approach to early requirements validation. A prototype is an enactable<sup>1</sup> mock-up or model of a software system that enables evaluation of features or functions through user and developer interaction with operational scenarios. Prototyping exposes functional and behavioral aspects of the system as well as implementation considerations, thereby increasing the accuracy of requirements and helping to control their volatility during development. The DSB report specifically recommends prototyping:

"In the decade since the waterfall model was developed, our discipline has come to recognize that setting the requirements is the most difficult and crucial part of the software building process, and one that requires iteration between the designers and users. In best modern practice, the early specification is embodied in a prototype, which the intended users can themselves drive in order to see the consequences of their imaginings. Then, as the design effort begins to yield data on the cost and schedule consequences of particular specifications, the designers and the users revise the specifications." [DSB87]

One example of the usefulness of validation via prototyping in real-time systems development is discussed in Bennett's "Modeling Radar Countermeasure Systems":

"When military intelligence identifies a new radar capability, a countermeasure is resultingly identified. Over time, the cycle of radar-threat identification, followed by countermeasure design, repeats itself. Each new set of ECM requirements becomes the starting point for additional engineering development. The particular requirements depend on the nature of the threat, and on the set of responses that might effectively be deployed. [...] The ECM system's real-time nature mandates that requirements analysis methods thoroughly support performance analysis and prediction. Whether the ECM is synchronous or asynchronous, the response's timing must be specified in such a way that its feasibility can be determined." [Bennett89]

---

<sup>1</sup>. An enactable model is one against which operational scenarios can be exercised in some automated fashion. Examples of model enaction include simulations, animations, mathematical dynamic analyses, and code execution.

The requirements of a system or a class of systems are gathered in an *evolutionary* fashion. Requirements knowledge is never complete, but rather evolves over time as new requirements are identified, existing requirements are expanded, and obsolete requirements are discarded (Figure 2-3).

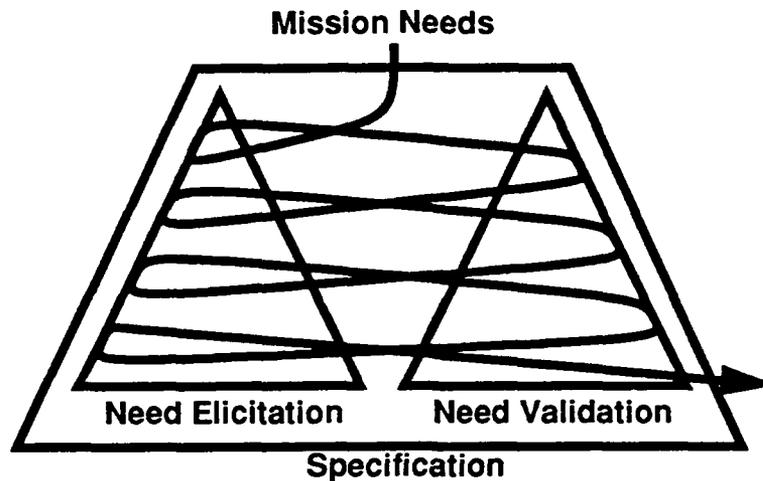


Figure 2-3 Evolving Mission Needs Into Validated Requirements

While requirements are gathered in an evolutionary and incremental fashion, product development is usually managed in a stepwise fashion. The widespread use of variations of the waterfall process model is indicative of the tendency of managers and developers to think in terms of idealized discrete phases, such as Requirements Analysis, High-Level Design, Detailed Design, Coding, Integration, and so forth. Strict adherence to such a process can be considered to be in *direct conflict* with the notion of evolutionary requirements engineering [Boehm88], inasmuch as the delivery of a complete requirements specification prior to the initiation of software design presumes that requirements can be fully understood before design activities begin.

Although the waterfall approach seems inimical to evolutionary development, it is undeniable that for a given requirement or set of requirements, the basic activities of requirements engineering, design, and implementation do take place, generally in an orderly fashion. In other words, one cannot implement a solution before one considers potential solutions, and one cannot consider potential solutions until one has a handle on the nature of the needs to be addressed. Thus, the difficulty with the waterfall view is not the specific activities or their ordering, but rather in the attempt to apply those activities to the development of the entire system as a whole unit. If instead we consider the entire system to be a collection of requirements (perhaps tens of thousands of them), we can successfully apply the waterfall to each requirement or to approachable subsets of requirements. This approach is more supportive of the evolutionary understanding of volatile requirements. In doing so, we can view software development as a continuous sequence of activities that are closely interrelated (Figure 2-4).

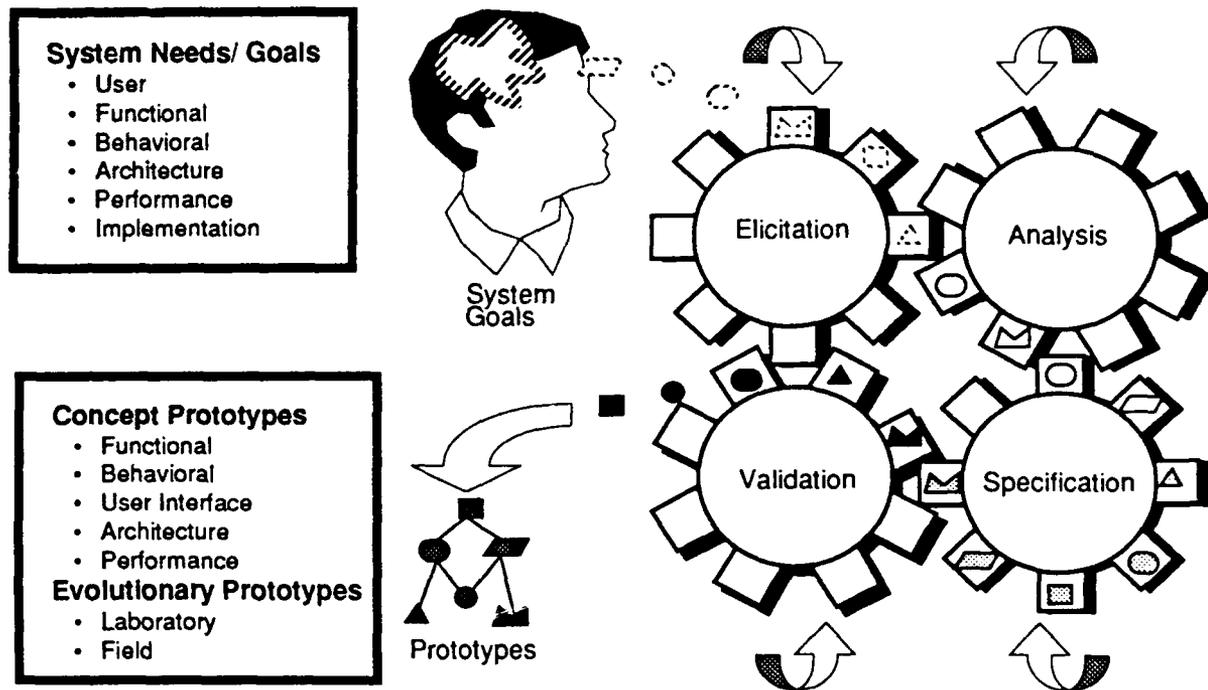


Figure 2-4 Prototyping as a Continuous Process

## 2.2.1 Classifying Prototyping Approaches

### 2.2.1.1 A General Purpose Taxonomy

The literature is replete with descriptions of software development and validation approaches that are referred to generically as prototyping, yet little consensus seems to exist regarding the exact process of creating and using prototypes. Sage and Palmer's "Software Systems Engineering" identified no less than eighteen prototyping approaches described by six different authors, and also showed the high degree of commonality among classes of approaches by mapping the different approaches into three classes [Sage90]. Because it is difficult to distinguish between the many proposed approaches based on a simplified classification scheme, we have provided a less generalized taxonomy in Table 1. Using this table, a given prototyping approach may be classified based on clear distinguishing characteristics rather than relatively ambiguous abstractions. Table 1 identifies five classification categories and twelve subcategories that help to determine the nature of the various approaches.

### 2.2.1.2 Two Prototyping Perspectives

Another useful way to classify prototyping approaches is according to the intended use and users of the prototype. In this sense, there are really two broad categories of prototyping approaches: those that involve the creation of a series of fielded prototypes, and those intent on exploring ideas without resorting to field deployment. The former are most commonly referred to as *field* or *evolutionary* prototypes, while the latter go by many names, including *rapid*, *con-*

*cept, throw-away, experimental, and exploratory* prototypes. For convenience, we will refer to these broad categories as **evolutionary** and **concept** prototypes respectively.

In essence, **concept prototyping** is a mechanism for achieving validation prior to commitment. Concept prototyping may be used to validate requirements prior to commitment to specific designs. Similarly, concept prototyping may be used to validate potential designs prior to commitment to specific implementations. In this sense, prototyping as a software development paradigm can be seen as tacit acceptance that requirements are not fully known or understood prior to design and implementation. Concept prototyping can be used as a means to explore new requirements and thus assist in the ongoing evolution<sup>1</sup> of requirements.

Viewed from a different perspective, the entire lifecycle of a product can be seen as a series of increasingly detailed **evolutionary prototypes**. Traditionally, the lifecycle is divided into two distinct phases: development and maintenance. Experience has shown that this distinction is somewhat arbitrary and betrays the reality that much or most of the cost of the software product lifecycle occurs after the product has been delivered [Lientz80]. This traditional viewpoint leads to numerous difficulties related to multiple perspectives on the concept of *quality*. For example:

- to a developer, a quality product is one that functions properly according to specification
- to a maintainer, a quality product might be considered one that readily lends itself to modification and enhancement
- to a user, a quality product is one that has the "right" look and feel, performance, and behavior
- to a customer, each of the above views are important aspects of quality

Because of this fundamental disparity in perceptions, it is not unusual for the maintaining organization to spend considerable time and effort in understanding the design and implementation of the product at hand [Lientz80], even while the resulting products do not meet user expectations. Further, the tendency to consider maintenance as a separate (and subsidiary) activity from development often results in severe underestimation of lifecycle costs.

The evolutionary view of the software lifecycle considers the first delivery to be an initial fielded prototype. Subsequent modifications and enhancements result in delivery of further, more mature prototypes. This process continues until eventual product retirement. Adoption of this view eliminates the arbitrary distinction between developers and maintainers, resulting in an important shift in mindset affecting strategies for cost estimation, development approach, and product acquisition.

---

<sup>1</sup>. Note that evolution in this context refers specifically to the evolution of *requirements*. By contrast, *evolutionary prototyping* refers to the use of prototypes for the evolution of a fielded *system*.

## 2.2.2 A Generic Prototyping Process

While the purposes and detailed processes of concept and evolutionary prototyping differ, there is a common abstract process that encompasses both views (Figure 2-4). In either case, there is a tightly interwoven sequence of process steps by which a set of system goals are transformed into an enactable model. System goals in the form of requirements or design constraints must be elicited from the originating source (e.g., customers, users, documentation). Elicited goals must be analyzed for various properties such as consistency and completeness, and subsequently codified in some form of specification (e.g., textual, graphical, mathematical). This specification then can form the basis for validation via prototyping or some other means. While this generic process is intuitively simple, a number of issues are worthy of note:

- As implied by Figure 2-4, each of the process steps drives and is driven by the other process steps. In other words, the elicitation of new needs and goals drives the analysis process, and by extension also the specification and validation processes. Similarly, the process of validation will lead to the elicitation of new or refined needs and goals.
- The individual process steps might range in formality from ad-hoc to highly formal.
- During the entire product lifecycle, the number of iterations of the sequence of activities from elicitation through validation will be very high, perhaps in the thousands or tens of thousands.<sup>1</sup>
- During any given iteration, any of the process steps might approach zero in terms of time or effort expended. For example, a well-understood subset of requirements from a precedented domain might require little or no significant analysis once those requirements have been elicited. Further, their specification might be recorded informally, and validation might consist of an informal confirmation by the customer of the specification.

The interwoven, continuous sequence of process steps described by Figure 2-4 apply equally well to any type of system needs, including user (high-level), functional, behavioral, and user interface requirements, and also design and implementation constraints on architecture and performance. Both concept and evolutionary prototyping can be used as validation mechanisms supporting this process view.

---

<sup>1</sup> A fallacy of the *waterfall* view is that the sequence of activities occur in a single iteration.

### 2.2.3 Common Prototyping Activities

In addition to the generic abstract process discussed in the preceding section, the various prototyping approaches share a common set of high-level activities (Figure 2-5). Regardless of

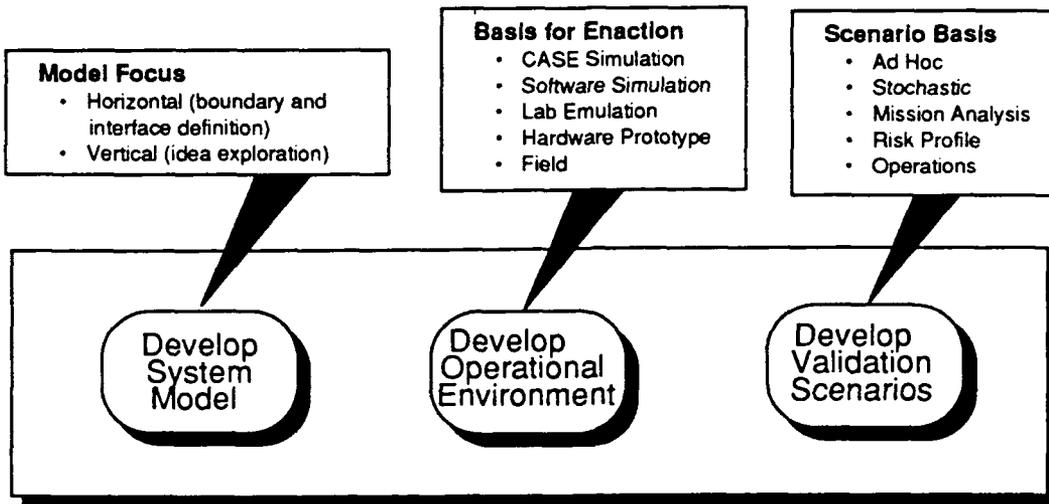


Figure 2-5 Common Prototyping Activities

the prototyping approach used, it is necessary to:

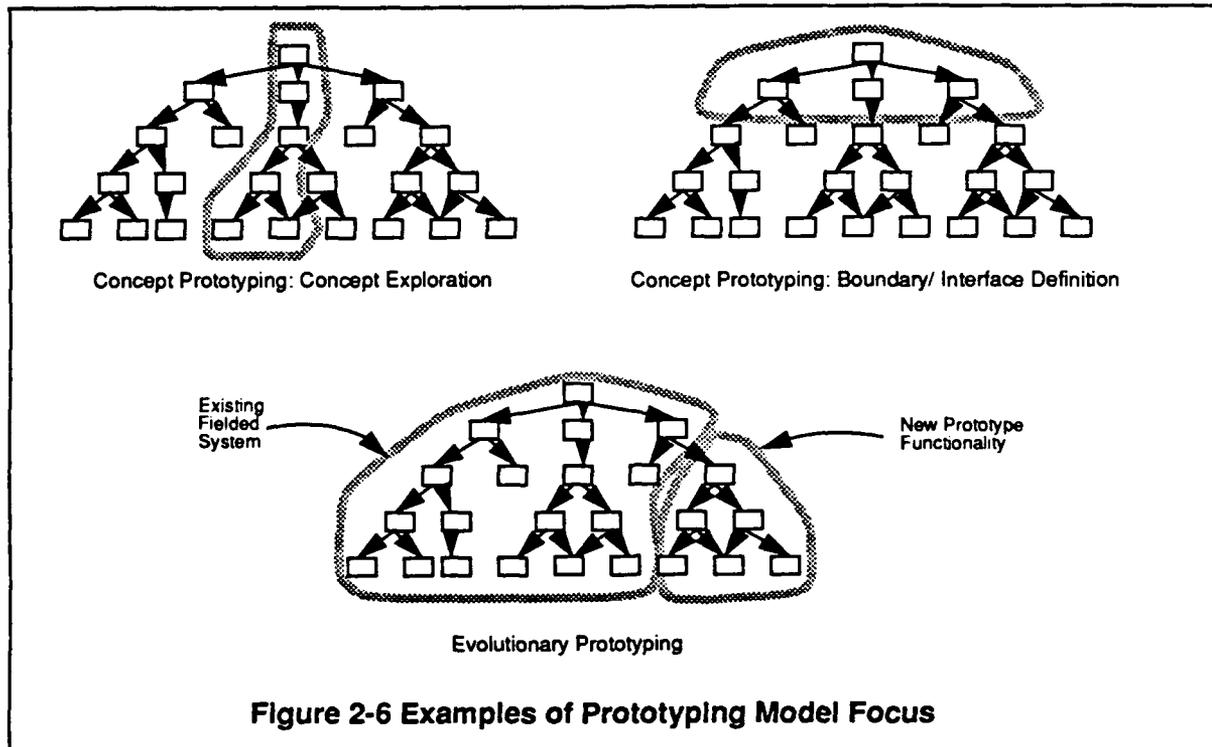
- create an enactable model of the system under development
- create an environment in which the system model will operate
- create a validation suite comprised of operational scenarios

The system model may exhibit varying amounts of breadth and depth of detail depending on the purpose of the prototype (Figure 2-6). For example, one might use a concept prototyping approach to explore the feasibility of certain requirements (i.e., a vertical slice) or to validate the boundaries and interfaces of the proposed system (i.e., a horizontal slice). An evolutionary prototype might involve the enhancement of an existing fielded system by insertion of new prototype functionality suitable for user exploration.

The environment in which the system model will operate may be the actual field environment in which the final product will operate (particularly for evolutionary prototypes), or the environment may consist of combinations of software simulation environments and hardware prototypes. An example of the latter would be a test and evaluation lab for a flight control system, where the prototype software is embedded within prototype hardware and operates in an environment consisting of real-time sensor simulators.

The validation suite may be derived in a variety of ways depending upon the intent of the prototype. For example, scenarios might be generated on an ad-hoc basis if the intent is to allow the users to become acquainted with a proposed user interface at their own convenience and

pace. By contrast comprehensive functional validation requires a more formal approach to creation of the validation suite.



## 2.2.4 Different Approaches for Different Needs

As discussed in the preceding sections, the generic process depicted in Figure 2-4 and activities depicted in Figure 2-5 apply to both the concept prototyping and evolutionary prototyping approaches. While recognition of this abstract commonality is useful, it is also important to recognize that there are significant differences among the various prototyping methods. The taxonomy presented in Table 1 highlights several general categories for recognizing these differences among *classes* of prototyping approaches. As a matter of practicality, it is important to consider specific approaches from the perspective of the types of needs that they address.

Figure 2-7 depicts one possible mapping between types of needs and potential validation methods. For example, one might apply a user interface mock-up or hypermedia approach to produce workable prototypes for validation of user-level requirements. If the intent is to validate functional and behavioral requirements that involve a much greater level of detail than user-level requirements, one might use an executable specification approach that provides an interactive animation capability. Laboratory simulation or analytical methods might form a good basis for validation of architectural and performance requirements. Finally, successive fielded prototypes might provide the best mechanism for validation of detailed implementation constraints upon the system under development. The salient point is that the perspectives of

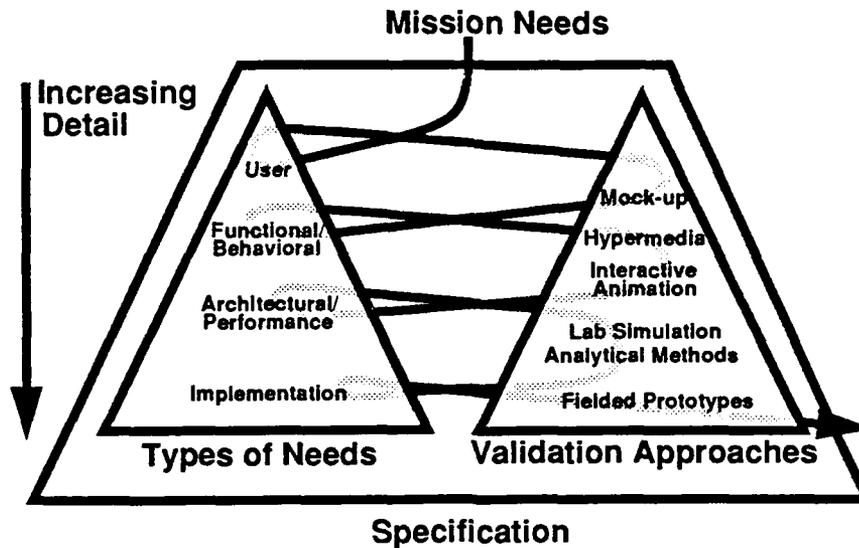


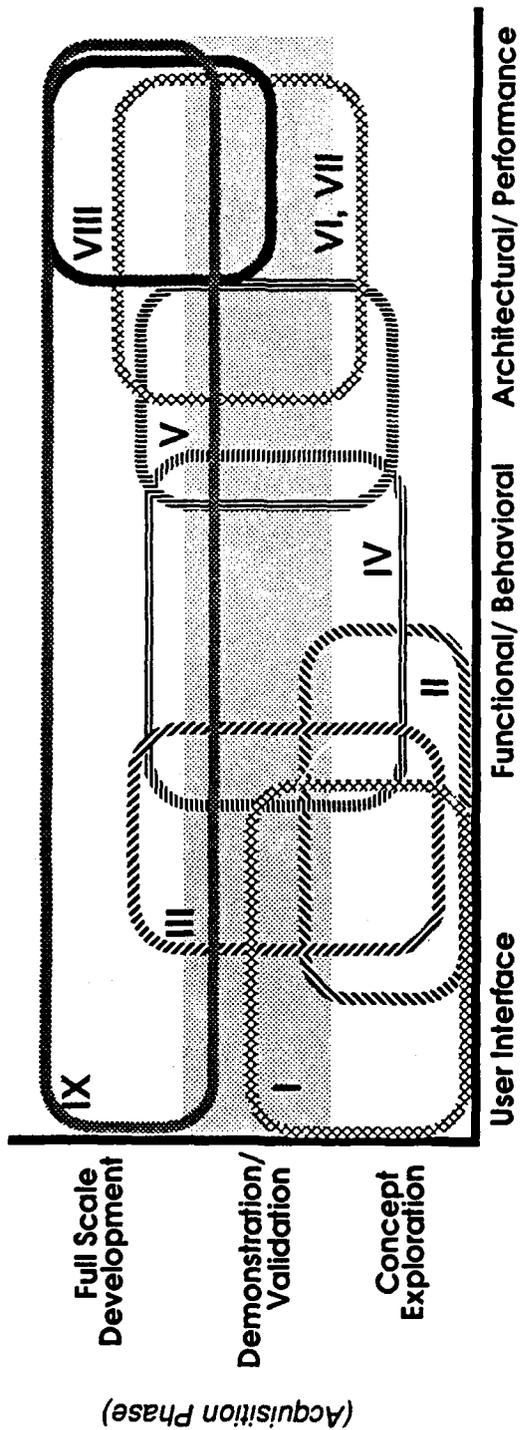
Figure 2-7 Different Approaches for Different Needs

concept prototyping and evolutionary prototyping are orthogonal and their purposes are complimentary. It should be possible to select a set of prototyping techniques based on specific needs and to apply those techniques in a complimentary fashion.

### 2.3 A Framework for Method and Tool Selection

While Figure 2-7 provides a high-level mapping of validation strategies for specific needs, it should not be taken to indicate that only one strategy is applicable or useful for a given development effort. In fact, because a project will exhibit varying needs during its lifecycle, it is likely that the development effort can benefit from a selection of several prototyping techniques. Section 2.2 indicated that the two broad categories of prototyping approaches, concept and evolutionary prototyping, have a common process in the abstract but have somewhat different purposes. A closer examination of these broad categories reveals that they are in fact highly complimentary.

Figure 2-8 provides a sample framework for the selection of prototyping methods and tools. The matrix indicates approximate coverage areas of various methods and tools against two dimensions, *Acquisition Phase* and *Validation Focus*. Other dimensions are possible, and certainly additional methods, tools, and categories could be added; however the figure clearly illustrates the complimentary nature of concept prototyping (Types I through VIII) and evolutionary prototyping (Type IX).



(Validation Focus)

Type	Technique for prototyping...	Method/ Tool Examples
I	Interactive graphical interface	Serpent, Rapid Prototyping System/REE, Hypercard
II	Control Panel/ Mock-up	Hypercard, Foresight, StateMate
III	Environment Interface/ Dynamic Visualization	Foresight, StateMate (interactive I/O, panel editor)
IV	Functional/Behavioral requirements	CAPS, Cruise, Proto+, Foresight, StateMate, KBSA
V	Logical allocation to subsystems	Parallel Proto, Foresight, StateMate, KBSA
VI	Physical distributed processing architecture	DURRA, Parallel Proto
VII	Task prioritization and allocation	Rate Monotonic Analysis, CAPS
VIII	Platform evaluation, low-level experimentation	Target processors, operating systems, and languages
IX	Evolutionary development/ incremental product delivery	Target processors, operating systems, and languages

Figure 2-8 A Prototyping Technology Framework

### 3 Summary and Directions

Poor understanding of system requirements, and by extension the lack of adequate validation of the correctness of requirements, has been discussed as a major cause of system failure and customer dissatisfaction. It has been proposed that an underlying cause of requirements difficulties is a reliance on idealized software lifecycle models that fail to take into consideration the fact that requirements evolve over time. It has been suggested that the adoption of lifecycle models and development paradigms that exploit the evolutionary nature of requirements may resolve requirements problems. Examples of these technologies include iterative and spiral lifecycle models and continuous prototyping paradigms.

Two major prototyping approaches have been identified: concept and evolutionary prototyping. Concept prototyping involves the exploration of new concepts, usually by experimentation in an isolated non-field environment and often through the use of simulation, mock-up, or 4th Generation languages. Evolutionary prototyping involves the iterative augmentation of a fielded system with new capabilities that have been developed without the usual developmental constraints in order to emphasize rapid deployment for prototype experimentation.

It has been suggested that both concept and evolutionary prototyping are indicative of a continuous, iterating, tightly-integrated process of elicitation, analysis, specification, and validation, and that both kinds of prototyping involve certain fundamental activities, such as model development, operational environment development, and scenario development.

Examination of some of the various proposed prototyping techniques and tools has revealed that there are many sub-categories of prototyping approaches. Some approaches are more suitable for validation of specific types of requirements than others. Similarly, some forms of prototyping may be more useful during certain acquisition phases than others. A framework has been proposed to highlight some of the relationships between prototyping techniques. An expanded framework should prove to be a useful aid for selection of a set of prototyping techniques that best meet the unique needs of particular programs and organizations.

Significant work remains in both the extension and elaboration of the proposed framework and in the examination of and experimentation with specific prototyping techniques with a goal of integration of requirements engineering and validation technology with other software engineering techniques. For example, the creation and use of domain-based specification libraries in support of rapid prototype generation will help to further establish engineering discipline for software development.

## 4 Taxonomy and Classification

Two tables are provided as guidelines illustrating the breadth of prototyping technology. Table 1 presents a taxonomy for classifying software prototyping approaches. Many authors have analyzed prototyping technology and attempted to provide a means for describing different approaches, creating unique terminology in the process. Table 1 attempts to provide a more atomic list of categories against which the proposed taxonomies of previous authors may be compared and contrasted. In this way, our taxonomy can help to identify areas of both redundancy and uniqueness. This table provides the following categories:

1. Phase - orientation of the approach in terms of requirements exploration/ definition or architecture and design.
2. Purpose - orientation in terms of learning/ eliciting requirements, validation of requirements, or exploration of feasibility or design alternatives.
3. Process - orientation toward concept or evolutionary prototyping.
4. Platform - model enactment occurs in a CASE or laboratory simulation environment or in a field environment.
5. Format - development of model accomplished with "4th Generation" languages, simulation languages, or target languages.

For the most part, the technique classes discussed in Table 1 do not specify platforms or formats, but these categories should prove useful in classifying specific techniques in each class.

Table 2 presents a comprehensive classification of literature found in the annotated bibliography (see **Section 5, "Bibliography"**). This table classifies the content of each bibliography entry in terms of purpose, techniques espoused, paradigms described, domain orientation, and prototyping process. These classifications, combined with the cross-indices provided at the end of this report, should help readers to rapidly locate literature relevant to their particular needs. Short descriptions of each category follow:

- Purpose
  - Learning/ Elicitation - identification of user needs, understanding of operational context, understanding of interface with other components.
  - Requirements Validation - user or customer confirmation of correctness and appropriateness of requirements, design, or implementation.
  - Alternatives Evaluation - examination of multiple scenarios for possible requirements variants.
  - Feasibility Assessment - determine and demonstrate the capability, suitability, or reasonableness of particular solutions to defined problems.
  - Design Issues Comprehension - explore design issues and alternatives to deepen understanding of decision implications.
- Techniques

- Mock-ups - prototypes are created as models presented in a format that is immediately recognizable to the users; normally, a mock-up is a user interface model that may or may not be skeletal in terms of functional capability.
- Executable Specifications - prototypes are created through the use of specification languages that are directly machine-interpretable.
- 4th Generation Languages - prototypes are developed by using a high-level language that is application-oriented; a 4th Generation Language is usually highly parameterized, allowing creation of an application through the selection of specific feature parameters.
- Module Construction - prototyping techniques that provide mechanisms and environments for creation, instantiation, interconnection, and use of reusable modules.
- Design and Coding - prototypes are created through traditional design and coding in the target programming language.
- Graphical - prototypes are created using a primarily graphical modeling language, such as mock-up editors or executable graphical specifications.
- Textual formalism - prototypes are created using a primarily textual modeling language, such as formal executable specifications or target code.
- Paradigm
  - Object-oriented - model orientation is object-oriented.
  - Functional - model orientation is based on functional components or data flow.
  - Knowledge-based - model is based on knowledge-assisted creation of prototypes.
- Domain Type
  - Database - domain is database/ transaction processing management information systems.
  - Real-time - domain involves real-time and embedded systems.
  - Distributed - domain involves distributed processing over multi-processor, local, or wide-area networks.
  - User Interface - primarily oriented toward user interface development.
  - Knowledge-base - prototyping approach is oriented toward development of knowledge-based systems.
- Process
  - Concept Prototyping - approach involves exploration of specific concepts where the prototype will not result in a fielded system.
  - Evolutionary Prototyping - approach involves initial use of prototypes that will be evolved eventually into a fielded system.
- Miscellaneous
  - Survey/ Collection - literature represents a technology survey or a collection of papers related to the subject of prototyping.
  - Roadmap - literature represents a useful part of a basic roadmap through prototyping technology.
  - Support Tool/ Environment - literature identifies or discusses an approach dependent upon one or more specific tools or environments.

Author	Classification	Phase	Purpose	Process	Platform	Format
		Requirements Architecture/ Design	Learn/Elicit/Understand Validate	Explore feas./ alternatives	Concept Prototyping Evolutionary Prototyping	Simulated Field
Sage/ Palmer	structural	✓		✓		✓
	functional	✓	✓			
	purposeful	✓	✓			
Church et al	explore reqs.	✓	✓			
	invest. altern.		✓	✓		
	feasibility dem.		✓	✓		
Riddle/Williams	evolutionary	✓			✓	✓
	experimental	✓	✓	✓		
	exploratory	✓	✓			
Carey/Mason	Version 0	✓	✓	✓	✓	✓
	demonstration	✓	✓	✓	✓	✓
	scenario	✓	✓	✓	✓	✓
Hekmatpour	incremental	✓	✓	✓	✓	✓
	evolutionary	✓	✓	✓	✓	✓
	throw-away	✓	✓	✓	✓	✓
Schneider	concept	✓	✓	✓		
	laboratory		✓	✓	✓	✓
	field		✓	✓	✓	✓
Freeman	decision		✓	✓	✓	
	preliminary		✓	✓	✓	✓
	concurrent	✓		✓	✓	
STEP	concept expl.	✓	✓	✓		
	solution eval.		✓	✓		✓
	devel. eval.		✓		✓	✓

**Table 1 Prototyping Approach Taxonomy**

Reference ID	Purpose					Techniques					Paradigm			Domain Type			Process			Misc.					
	Learning/Elicitation	Requirements Validation	Evaluate Alternatives	Assess Feasibility	Understand Design Issues	Mock-Ups	Executable Specifications	4th Generation Languages	Module Construction	Design and Coding	Graphical	Textual Formalism	Object-Oriented	Functional	Knowledge-based	Database	Real-time	Distributed	User Interface	Knowledge-base	Concept Prototyping	Evolutionary Prototyping	Survey/Collection	Roadmap	Support Tool/ Environment
Andriole89	✓	✓	✓	✓		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Andriole90	✓	✓	✓	✓		✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Andrulis90													✓												
Bagrodia90			✓		✓		✓	✓								✓	✓					✓			
Bajwa89					✓				✓													✓			✓
Bailes87														✓	✓										
Barbacci91					✓		✓						✓	✓			✓	✓							
Bartschi89					✓			✓		✓			✓			✓			✓						
Berardi89				✓	✓				✓					✓			✓								✓
Biggie89	✓	✓	✓			✓															✓				
Birch89	✓										✓			✓		✓									
Black91	✓	✓														✓	✓		✓			✓	✓	✓	✓
Brooks87																					✓	✓		✓	✓
CASEOut90																							✓	✓	✓
Chang90			✓	✓	✓		✓	✓		✓		✓	✓			✓					✓			✓	✓
Cohen90			✓	✓	✓		✓	✓		✓		✓	✓			✓					✓			✓	✓
Cooling89																✓						✓	✓		✓
Cooper90		✓											✓					✓							✓
Cordy91				✓							✓								✓			✓			✓
DeSoi89		✓	✓							✓				✓				✓							✓
Degl'Innocenti90							✓				✓					✓									
Demeure89					✓					✓						✓	✓								✓
Diaz-Gonzalez91		✓									✓	✓				✓									✓
Edmonson89	✓	✓	✓					✓		✓								✓			✓				✓
Ekambareshwar89		✓									✓		✓												
Espinosa90				✓							✓					✓									✓
Fisher87	✓	✓						✓							✓			✓			✓		✓		✓
Ganti90			✓	✓					✓			✓									✓				✓
Gerber90									✓				✓			✓									
Gimnich87		✓	✓		✓						✓		✓												

Table 2 Prototyping Literature Classifications

Reference ID	Purpose					Techniques					Paradigm			Domain Type			Process		Misc.							
	Learning/Elicitation	Requirements Validation	Evaluate Alternatives	Assess Feasibility	Understand Design Issues	Mock-Ups	Executable Specifications	4th Generation Languages	Module Construction	Design and Coding	Graphical	Textual Formalism	Object-Oriented	Functional	Knowledge-based	Database	Real-time	Distributed	User Interface	Knowledge-base	Concept Prototyping	Evolutionary Prototyping	Survey/Collection	Roadmap	Support Tool/ Environment	
Giordano91	✓	✓			✓	✓			✓						✓						✓					
Gomaa81	✓	✓							✓							✓						✓			✓	
Gonzalez89							✓						✓													
Gregorio90		✓				✓				✓			✓			✓	✓		✓							✓
Gutierrez89													✓										✓	✓		✓
Harris87	✓	✓	✓												✓				✓			✓				
Hartson91		✓	✓			✓													✓							
Hawryszkiewicz87					✓			✓							✓							✓	✓			
Heisler88													✓													
Hekmatpour90				✓	✓								✓		✓									✓		
Henskes87										✓									✓			✓				
Henskes90					✓					✓			✓						✓			✓				✓
Hughes89	✓	✓														✓										
IEE89				✓												✓										
Jain89			✓							✓			✓										✓			
Jones90		✓				✓												✓								✓
Koch88					✓					✓					✓											
Konrad	✓	✓				✓				✓	✓		✓													✓
Kordon90					✓					✓						✓						✓				
Kozubal90					✓					✓					✓	✓	✓					✓				
Kreutzer90a										✓				✓												✓
Kreutzer90b										✓				✓												✓
Krista89		✓								✓				✓												
Lea90		✓					✓						✓													✓
Luckey90				✓	✓					✓			✓		✓			✓								
Luqi86	✓	✓	✓	✓	✓			✓		✓	✓		✓			✓					✓	✓		✓	✓	
Luqi87	✓	✓	✓	✓	✓			✓		✓	✓		✓			✓					✓	✓		✓	✓	
Luqi89	✓	✓	✓	✓	✓			✓		✓	✓		✓			✓					✓	✓		✓	✓	
Luqi90a	✓	✓	✓	✓	✓			✓		✓	✓		✓			✓					✓	✓		✓	✓	
Luqi90b	✓	✓	✓	✓	✓			✓		✓	✓		✓			✓					✓	✓		✓	✓	

Table 2 Prototyping Literature Classifications

Reference ID	Purpose					Techniques					Paradigm			Domain Type		Process		Misc.							
	Learning/Elicitation	Requirements Validation	Evaluate Alternatives	Assess Feasibility	Understand Design Issues	Mock-Ups	Executable Specifications	4th Generation Languages	Module Construction	Design and Coding	Graphical	Textual Formalism	Object-Oriented	Functional	Knowledge-based	Database	Real-time	Distributed	User Interface	Knowledge-base	Concept Prototyping	Evolutionary Prototyping	Survey/ Collection	Roadmap	Support Tool/ Environment
Luqi91	✓	✓	✓	✓	✓			✓		✓	✓		✓			✓				✓			✓	✓	
Luqi92	✓	✓	✓	✓	✓			✓		✓	✓		✓			✓				✓			✓	✓	
Madison89			✓								✓			✓					✓				✓	✓	
Matsumoto																							✓		
McEnery90		✓						✓							✓				✓					✓	✓
McInroy89					✓			✓	✓					✓										✓	✓
Minkowitz89			✓	✓	✓						✓	✓												✓	✓
Nugent88					✓						✓	✓												✓	✓
O'Neil89		✓						✓			✓		✓		✓										✓
Ortner88								✓							✓										✓
Overmyer90	✓	✓																			✓		✓		✓
Powers89					✓														✓						✓
Purtilo91				✓				✓			✓					✓									✓
Rizman90a										✓	✓		✓							✓				✓	✓
Rizman90b										✓	✓		✓							✓				✓	✓
Royce89			✓	✓	✓			✓	✓						✓			✓							✓
Rzepka86		✓	✓	✓				✓	✓						✓			✓			✓				✓
SEKE90												✓		✓											✓
Shirota89			✓		✓			✓											✓						✓
Smith90					✓			✓							✓			✓							✓
Smyrniotis90a	✓	✓						✓						✓							✓				✓
Smyrniotis90b	✓	✓						✓						✓							✓				✓
Son88					✓			✓	✓						✓			✓							✓
Tamanaha90			✓		✓			✓	✓		✓		✓		✓			✓			✓	✓			✓
Tenazas90		✓	✓					✓	✓						✓						✓				✓
Thayer90																							✓	✓	
Trenouth91			✓	✓	✓				✓												✓				✓
Tsai89					✓			✓			✓	✓	✓	✓											✓
Tsai90a		✓			✓			✓		✓	✓	✓	✓	✓											✓
Tsai90b		✓			✓			✓		✓	✓	✓	✓	✓											✓

Table 2 Prototyping Literature Classifications

Reference ID	Purpose				Techniques				Paradigm			Domain Type		Process		Misc.									
	Learning/ Elicitation	Requirements Validation	Evaluate Alternatives	Assess Feasibility	Understand Design Issues	Mock-Ups	Executable Specifications	4th Generation Languages	Module Construction	Design and Coding	Graphical	Textual Formalism	Object-Oriented	Functional	Knowledge-based	Database	Real-time	Distributed	User Interface	Knowledge-base	Concept Prototyping	Evolutionary Prototyping	Survey/ Collection	Roadmap	Support Tool/ Environment
Tucherman90					✓					✓				✓	✓										✓
Turnheim89						✓								✓					✓						✓
Tyszberowicz89																									
Tyszberowicz91a		✓	✓	✓		✓				✓		✓				✓					✓				
Tyszberowicz91a		✓	✓	✓		✓				✓		✓				✓					✓				
Wallentinson89					✓														✓		✓				✓
Warkowski90					✓			✓													✓	✓			✓
Wellner89	✓				✓					✓									✓		✓				✓
Whatmore91		✓	✓	✓				✓				✓				✓					✓				
Wing91					✓					✓	✓		✓									✓			
Zhao91					✓						✓								✓						✓
Zompi90					✓	✓				✓		✓				✓	✓								✓

Table 2 Prototyping Literature Classifications



## 5 Bibliography

This section presents an annotated bibliography of references on **prototyping** technology for the development of computer-based systems. In addition to the appropriate publication information for each document type, all bibliographic citations include either 1) an *abstract* taken from the document itself, usually written by the author, or 2) an *annotation* written for the document by the compilers of this bibliography. In most cases, the author's abstract was used. Annotations are provided in those instances where the document did not include an abstract, or where the abstract was considered insufficient. These abstracts and annotations should help the reader to determine if there is interest in a given citation. Various indices are also included to simplify the task of locating a particular reference or a range of articles in a subject area.

### 5.1 Bibliography Selection Criteria

The amount of available literature describing software system prototyping is enormous and continues to grow. Due to this growth, it is impossible to call any bibliography "complete" by the time it is published; in fact, it is unlikely to include every relevant publication while the bibliography is being researched. In order to produce a relevant yet manageable bibliography, the authors applied the following selection criteria to each potential reference:

- key words and phrases used for the search included "software" and "rapid prototyping"
- primary references were limited to those dating from 1988 to present
- some additional references were included as seminal historical references upon which later work in prototyping was based

Some otherwise relevant citations were excluded from the bibliography if they were:

- proprietary to an organization
- copies of slide presentations (which are typically unpublished and difficult to understand without the accompanying talk)
- especially difficult to obtain, or
- superseded by later, better defined work of the same author(s)

If an appropriate publication has been omitted from this bibliography, please contact the authors at:

Requirements Engineering Project  
Software Engineering Institute  
Carnegie Mellon University  
Pittsburgh, PA 15213-3890

## 5.2 Annotated References

- Identifier:** Andriole89  
**Title:** *Modern Life Cycling: Some System Design Principles for the 90s*  
**Author:** Andriole, S.J.  
**Source:** AFCEA International Press, Fairfax, Virginia  
**Date:** 1989  
**Keywords:** process, methods, survey  
**Notes:** The author makes a case for far-ranging re-examination of the way systems are engineered, including (1) the need for multidisciplinary information systems engineering, (2) the need for flexible life cycle methods and models, and (3) renewed commitment to multidisciplinary education, training, and research. Communication with models and prototypes is emphasized, particularly conceptual prototypes. Calls for elimination of existing life cycle models in favor of a "situational life cycling" approach.
- Identifier:** Andriole90  
**Title:** *Command and Control Information Systems Engineering: Progress and Prospects*  
**Author:** Andriole, S.J.  
**Source:** Advances in Computers, Vol. 31, Academic Press, Inc.  
**Date:** 1990  
**Keywords:** process, methods, survey
- Identifier:** Andrullis90  
**Title:** *Object-oriented Development Aids Prototyping and Delivery*  
**Author:** Andrullis, M. W.  
**Source:** Signal, Volume 45, No. 4, pp. 76-8  
**Date:** December, 1990  
**Keywords:** object-oriented, Ada, process, survey  
**Abstract:** The author explains how [object-oriented] development seeks to strike a balance between the rigid, stylized and incremental waterfall life cycle and the loosely defined rapid prototyping approach. She commends its combination with Ada software as an aid to prototyping, reuse and delivery. Managers relying on the combination enjoy the visibility of a system view and can better understand and control progress against schedule while ensuring better software engineering.

**Identifier:** Bagrodia90  
**Title:** *An Integrated Approach to the Design and Performance Evaluation of Distributed Systems*  
**Author:** Bagrodia, R.L.  
**Source:** Systems Integration '90 Proceedings; pp 662-71; IEEE Computer Society Press  
**Date:** April, 1990  
**Keywords:** real-time, distributed processing, methods  
**Abstract:** Iterative transformation of performance models into operational systems is a desirable goal for rapid prototyping technologies. The author describes an approach to transforming a hybrid model into an operational system that can be shown to satisfy its performance specifications. A hybrid model merges operational modules with abstract simulation modules. The capabilities supported by the methodology include: (a) interrupt handling, (b) integration of distributed software and hardware components, and (c) evaluation of the effect of upgrading existing hardware components. The author also describes how simulation algorithms can be adapted to execute hybrid modules. The PIPS (partially implemented performance models) methodology can handle interruptible processes and include the execution of hybrid models in which the physical module executes faster than the logical module.

**Identifier:** Bailes87  
**Title:** *Software Development by Functional Language Prototyping*  
**Author:** Bailes, P. A.; Salzman, E. J.  
**Source:** Technical Report, University of Queensland Department of Computer Science  
**Date:** 1987  
**Keywords:** formalism

**Identifier:** Bajwa89  
**Title:** *Lessons Learned from the Use of a Spiral Model for an Ada Development Effort: the Software Life Cycle Support Environment (SLCSE)*  
**Author:** Bajwa, L. Y.  
**Source:** Proceedings of the IEEE 1989 NAECON, May, 1989; pp.1807-12, vol. 4  
**Date:** 1989  
**Keywords:** applications: environment, process, Ada  
**Abstract:** A description is given of the software life cycle support environment (SLCSE), a VAX-based software development environment. The project was 80% Ada, subject to DOD-STD-2167 reviews and documentation. SLCSE adopted an incremental-build/rapid-prototyping methodology, similar to Boehm's spiral model. Eight prototype builds (and the final product) were delivered during the course of the development effort. At the heart of the system is an integrated database implemented via a commercial relational database management system overlaid with an entity-relationship layer that models the entire information content of DOD-STD-2167A. The lessons learned from this project are discussed. Among these lessons are the findings that incremental builds help to prove the viability of critical design approaches and that the use of a spiral-type model requires a new mindset.

**Identifier:** Barbacci91  
**Title:** *Durra: An Integrated Approach to Software Specification, Modeling and Rapid Prototyping*  
**Author:** Barbacci, M. R.; Lichota, R. W.  
**Date:** 1991  
**Keywords:** real-time, distributed processing, methods, executable specifications, structured analysis  
**Abstract:** Software specification, modeling and prototyping activities are often performed at different stages in a software development project by individuals who use different specialized notations. The need to manually interpret and transform information passed between stages can significantly decrease productivity and can serve as a potential source of error. The recent development of commercial executable specification tools represents a potential semi-automated link between specification, modeling and prototyping activities. Many of these tools use a graphical notation based on real-time structured analysis to represent software models and provide a built-in simulation capability. Unfortunately, these tools can be inadequate for analyzing the performance of complex real-time systems. Moreover, the prototypes generated from realistic software specification tend to be much too inefficient for these applications. We feel that to effectively link specification, modeling, and prototyping activities, integration must occur at the level of a technical architecture. This corresponds to a software architecture augmented to include formal descriptions of task behavior. We believe that Durra, currently under development at the Software Engineering Institute, can provide this integration. Durra is a non-procedural language designed to support the development of distributed applications consisting of multiple, concurrent, large-grained tasks executing in a heterogenous network. Durra provides a framework through which one can specify the structure of an application in conjunction with its behavior, timing, and implementation dependencies. These specification may be validated by passing behavioral and timing information associated with each Durra task description to a runtime interpreter. Similarly, software prototypes may be constructed by directing this information to a suitable source code generator. We have already developed an interpreter and source code translator for a language based on simple timing expressions. We are presently constructing a source code generator for a more complex language defined by SMARTS (the Specification Methodology for Adaptive Real-Time Systems) developed by Hughes Aircraft Company).

**Identifier:** Bartschi89  
**Title:** *Information Systems Design and Prototyping Using an Object-oriented Software Engineering Environment*  
**Author:** Bartschi, M.; Rieche, B.; Tresch, M.  
**Source:** TOOLS '89. Technology of Object-Oriented Languages and Systems Proceedings; pp. 423-36; Paris, France; 13-15 Nov. 1989 Paris, France; SOL; 1989; 591 pp.  
**Date:** November, 1989  
**Keywords:** methods, database, object-oriented, graphical specification, tools  
**Abstract:** A collection of methods is presented that allow the integration of all design steps of interactive information systems into a uniform software engineering environment. This methodology combines advanced database design techniques with object-oriented design methods and allows subsequent prototype generation on the basis of the graphical system specification. For this purpose, features for alphanumeric form management and for 2D graphical editing are combined in a unified user interface management system. Further, the architecture of PARADISE, an environment implementing these methods, is described and an application example given to show the methods' strengths.

**Identifier:** Berardi89  
**Title:** *Rapid Prototyping of Complex Avionics Systems*  
**Author:** Berardi, L.  
**Source:** AGARD Lecture Series 164. Systems Engineering (AGARD-LS-164); pp. 7/1-18; Kettering, OH, USA; 15-16 May 1989 Sponsored by: AGARD Neuilly sur Seine, France; AGARD; 1989; v+134 pp.  
**Date:** May, 1989  
**Keywords:** applications: avionics, methods, tools, knowledge-based  
**Abstract:** The use of a rapid prototyping approach in the initial stages of complex avionics system design can complement some traditional computer design methods. In fact most of the computer aids in engineering and design is aimed to a better, coherent and, as far as possible, complete description of the project, but not too much is done on the verification of the proposed concept implementation. The author discusses the advantages to have available during the early design stages a software prototype of the system to highlight undesirable characteristics or possible improvements when the system has a high degree of complexity. Then a design tool called ECATE (Expert Consultant for Avionics System Transformation Exploitation), developed by Avionics Systems Group of Aeritalia, is described. ECATE is an expert system that prototypes the information handling architecture of an avionics system. The use of knowledge engineering and, in general, artificial intelligence approach for the rapid prototyping has been proven very effective, because of the high flexibility, complex domain mastering capability, and heuristic methods typical of these techniques. A description of a complete, integrated environment for the rapid development of prototypes of avionics systems, by using artificial intelligence and computer tools, is given of the high flexibility, complex domain mastering capability, and heuristic methods typical of these techniques. A description of a complete, integrated environment for the rapid development of prototypes of avionics systems, by using artificial intelligence and computer tools, is given.

**Identifier:** Biggie89  
**Title:** *A Multimedia Rapid Prototyping Tool for the Development Of Computer-assisted Instruction*  
**Author:** Biggie, A.V.L.; Buchanan, W.E.; Hazan, P.L.; Kossiakoff, A.  
**Source:** Johns Hopkins APL Technical Digest; vol.10, no.3; July-Sept. 1989; pp. 246-53  
**Date:** September, 1989  
**Keywords:** tools, applications: education  
**Abstract:** A rapid prototyping tool has been designed to aid in the creation of computer-assisted instruction (CAI) software for children with learning disabilities and mental retardation. The tool, which was conceived and developed under the collaborative program between APL and the JHU Division of Education in the School of Continuing Studies, has enabled a multidisciplinary team of educators and computer engineers to visualize and test all features of proposed CAI programs on-line during regular design sessions held around a conference table. Computer program development time has thereby been significantly shortened, and significant gains have been made in the quality of educational products produced.

**Identifier:** Birch89  
**Title:** *A Knowledge Base Approach to the Specification Of Real Time System Requirements*  
**Author:** Birch, M.; Whiteley, K.  
**Source:** Second International Conference on Software Engineering for Real Time Systems (Conf. Publ. no.309); pp. 21-5; Cirencester, UK; 18-20 Sept. 1989 London, UK; IEE; 1989; xii+287 pp.  
**Date:** September, 1989  
**Keywords:** knowledge-based, real-time  
**Abstract:** The use of pattern recognition based on a Holland classifier produced a fast, efficient expert system with a malleable rule base. By decoupling the inference mechanism from the knowledge base it is a simple process to add new rules or extended the scope of the rules within the rule base. In addition, Smalltalk-80 provided a highly interactive rapid prototyping environment which, with its resident bit block transfer mechanism, is ideal for the fast manipulation of large character strings implemented as bitmap masks. By using breakpoints in the code at run-time, debug windows can be opened which provide such facilities as single-step execution and inspection of bitmaps using the bit editor. Such powerful debugging facilities greatly reduced the implementation time for the tool.

**Identifier:** **Black91**  
**Title:** *Acquisition Model for the Capture and Management of Requirements for Battlefield Software Systems*  
**Author:** Black, H.; Leciston, D.; McGhee, R.; Zimmerlich, J.  
**Source:** CECOM-TR-90-2  
**Date:** January, 1991  
**Keywords:** user-interface, real-time, process  
**Notes:** This report presents an acquisition model that meets the needs of new and unprecedented systems that are software intensive, large, complex, and have extensive man-machine interface requirements. When properly applied, it should reduce the cost, schedule, and quality risks that have been associated with these types of procurements. This model is proposed within the context of DOD-STD-2167A and can be tailored to apply to a wide range of acquisitions. This model acknowledges that requirements have not and perhaps can not be fully and adequately specified up front, prior to acquisition, especially for large and complex systems. Rather, they evolve throughout the system life cycle. It stresses that requirements must be engineered and managed, not merely written. The model proposes six risk reduction strategies, which have been previously recommended by numerous DoD studies. This report provides guidance for the project manager on their implementation.

**Identifier:** **Brooks87**  
**Title:** *No Silver Bullet: Essence and Accidents of Software Engineering*  
**Author:** Brooks, F. P. Jr.  
**Source:** IEEE Computer  
**Date:** April, 1987  
**Keywords:** process, methods  
**Abstract:** Fashioning complex conceptual constructs is the essence; accidental tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence.  
**Notes:** Brooks argues persuasively that requirements refinement and rapid prototyping is one of only a handful of technologies that hold promise for attacking the essential rather than accidental difficulties of software development (other cited technologies are: buy rather than build, develop incrementally, and develop great designers). This article is closely related to the [DSB87] report.

**Identifier:** **CASEOut90**  
**Title:** *Survey of Rapid Prototyping Tools*  
**Source:** CASE Outlook 90, No. 4  
**Date:** 1990  
**Keywords:** survey, tools  
**Abstract:** Rapid software prototyping, also known as Rapid Applications Development (RAD) continues to become increasingly visible as a CASE development strategy. Rapid prototyping success stories hold out the promise of applications that better match real-world needs, decreased development time, and responsive accommodation to changing requirements. Software prototyping still lies in the nether world between techniques, methodologies, and tools, but it is clearly a valuable approach in many situations.

**Identifier:** Chang90  
**Title:** *A Real-time Distributed Simulation of PBX with Software Reuse*  
**Author:** Chang, C.K.; Young-Fu Chang; Aoyama, M.  
**Source:** Simulation; vol.54, no.2; Feb. 1990; pp. 71-9  
**Date:** February, 1990  
**Keywords:** real-time, distributed processing, reuse, applications: telephony, simulation  
**Abstract:** Rapid prototyping has proven to be a promising way of accomplishing a feasibility study. It cuts costs and reduces the complexity of the development of real-time distributed systems. UICPBX is a simulator of private branch exchange (PBX), an important member in the family of telephone switching systems. It has been prototyped using the C language in a SUN workstation environment under UNIX. During the development of UICPBX, techniques of software reusability were applied. First, a software hierarchy with three layers (kernel, basic supporting and calling functions) was employed according to the functional characteristics of the application system. Secondly, a software library was established to provide an effective environment to support reuse of functions developed at the middle (basic supporting) layer of the software hierarchy.

**Identifier:** Cohen90  
**Title:** *Rapid Prototyping of Communications Protocols Using a New Parallel Language*  
**Author:** Cohen, D.M.; Guinther, T.M.; Ness, L.A.  
**Source:** Systems Integration '90. Proceedings of the First International Conference on Systems Integration (Cat. No.90TH0309-5); pp. 196-204; Morristown, NJ, USA; 23-26 April 1990 Sponsored by: IEEE; New Jersey Inst. Technol.; ACM; AT&T; Bell Commun. Res.; Gesellschaft fur Math. & Datenverarbeitung Los Alamos, CA, USA; IEEE Comput. Soc. Press; 1990; xvi+800 pp. ISBN 0 8186 9027 5  
**Date:** 1990  
**Keywords:** languages, executable specifications, applications:communications  
**Abstract:** A description is given of the L.0 language, a parallel, high-level executable specification language created for the design and implementation of software systems with inherent concurrency, such as communications protocols, services and networks. L.0 was explicitly designed to express coordination, simultaneity, and the hierarchical composition of systems from component subsystems. L.0 has been used to prototype communications protocols and services and to study network architectures and switching systems. The application of L.0 to the prototyping of a large portion of an experimental data communication services network architecture is discussed.

**Identifier:** Cooling89  
**Title:** *The Emergence of Rapid Prototyping as a Real-time Software Development Tool*  
**Author:** Cooling, J.E.; Hughes, T.S.  
**Source:** Second International Conference on Software Engineering for Real Time Systems (Conf. Publ. no.309); pp. 60-4; Cirencester, UK; 18-20 Sept. 1989 London, UK; IEE; 1989; xii+287 pp.  
**Date:** September, 1989  
**Keywords:** real-time, tools, applications: industrial, survey  
**Abstract:** It is demonstrated that there is both the need and the means for rapid prototyping to establish itself as a software development tool for real-time embedded systems. The authors describe the problems that software developers face and in particular the nature of embedded real-time systems. The aims of rapid prototyping stand out quite clearly against this background. With the emergence of new concepts in software engineering, such as computer aided software engineering and object-oriented design these aims are becoming increasingly realisable as practical methodologies and tools. The authors examine some of these tools which have particular relevance to the development of real-time systems. The analysis yields several useful pointers as to the elements required of a full scale rapid-prototyping system for use in commercial industrial projects. They also outline some projects which demonstrate both the benefits and the difficulties encountered when using rapid prototyping in real-time systems.

**Identifier:** Cooper90  
**Title:** *Between Man and Machine*  
**Author:** Cooper, S.  
**Source:** Systems International; vol.18, no.1; Jan. 1990; pp. 37-8, 40  
**Date:** January, 1990  
**Keywords:** user-interface, object-oriented, tools  
**Abstract:** The author discusses the need for man-machine interfacing tools (MMI tool). She emphasizes the importance of the object oriented approach to software development with its encapsulation, instantiation/classes, and inheritance components. Object orientation offers conceptual benefits extensibility, robustness, and rapid prototyping/modification. An example of an MMI development system incorporating the above technologies in the Graphical Modelling System (GMS) from Sherrill-Lubinski. It provides facilities for users to build their own graphic representations of data together with the interaction models necessary for the application. Using GMS the MMI is distributed to dedicated user workstations managing the data display and interpretation demands from several applications.

**Identifier:** Cordy91  
**Title:** *TXL: A Rapid Prototyping System for Programming Language Dialects*  
**Author:** Cordy, J.R.; Halpern-Hamu, C.D.; Promislow, E.  
**Source:** Computer Languages; vol.16, no.1; 1991; pp. 97-107  
**Date:** 1991  
**Keywords:** applications: programming languages  
**Abstract:** Describes a rapid prototyping system for extensions to an existing programming language. Such extensions might include new language features or might introduce notation specific to a particular problem domain. The system consists of a dialect description language used to specify the syntax and semantics of extensions, and a context sensitive syntactic transducer that automatically implements the extensions by transforming source programs written using them to equivalent programs in the original unextended language. Because the transformer is context sensitive, it is more powerful than traditional context-free preprocessors and extensible languages can be used to prototype language extensions involving significantly new programming paradigms such as object-oriented programming.

**Identifier:** DeSol89  
**Title:** *Graphical Specification of User Interfaces with Behavior Abstraction*  
**Author:** DeSoi, J.F.; Lively, W.M.; Shepard, S.V.  
**Source:** SIGCHI Bulletin; spec. issue.; May 1989; pp. 139-44 Conference on Human Factors in Computing Systems (CHI 89); Austin, TX, USA; 30 April-4 May 1989 Sponsored by: IEEE; ACM  
**Date:** May, 1989  
**Keywords:** user-interface, graphical specification, knowledge-based, methods, tools  
**Abstract:** The Application Display Generator (ADG) is a graphical environment for the design and implementation of embedded system user interfaces. It is a major component of the Graphical Specification Subsystem (GSS) in Lockheed's Express knowledge-based software development environment. ADG gives non-programmers simple and flexible methods for graphically specifying the presentation and behavior of embedded system user interfaces. In the ADG methodology arbitrary presentations are attached to abstract object behaviors. This approach makes it possible to provide unconstrained presentations, intelligent user support, rapid prototyping, and flexible facilities for composing complex objects.

**Identifier:** Degl'Innocenti90  
**Title:** *RSF: A Formalism for Executable Requirement Specifications*  
**Author:** Degl'Innocenti, M.; Ferrari, G.L.; Pacini, G.; Turini, F.  
**Source:** IEEE Transactions on Software Engineering; vol.16, no.11; Nov. 1990; pp. 1235-46  
**Date:** November, 1990  
**Keywords:** executable specifications, logic programming, real-time, formalism  
**Abstract:** RSF is a formalism for specifying and prototyping systems with time constraints. Specifications are given via a set of transition rules. The application of a transition rule is dependent upon certain events. The occurrence times of the events and the data associated with them must satisfy given properties. As a consequence of the application of a rule, some events are generated and others are scheduled to occur in the future, after given intervals of time. Specifications can be queried, and the computation of answers to queries provides a generalized form of rapid prototyping. Executability is obtained by mapping the RSF rules into logic programming. The rationale, a definition of the formalism, the execution techniques which support the general notion of rapid prototyping and a few examples of its use are presented.

**Identifier:** Demeure89  
**Title:** *Prototyping and Simulating Parallel, Distributed Computations with VISA*  
**Author:** Demeure, I.M.; Nutt, G.J.  
**Source:** Technical report. University of Colorado, Boulder. Dept. of Computer Science; CU-CS-450-89 Prototyping and simulating with VISA.  
**Date:** 1989  
**Keywords:** distributed processing, tools, simulation, graphical specification  
**Abstract:** Designing high performance distributed computations is a challenging task. In this paper, we describe VISA (VISual Assistant), a software tool to support the design, prototyping, and simulation of parallel, distributed computations. In particular, VISA is meant to guide the choice of partitioning and communication strategies for such computations, based on their performance. VISA uses ParaDiGM (Parallel Distributed computation Graph Model) as a basis for its graphical interface. VISA supports the editing of ParaDiGM graphs, and the animation of these graphs to provide visual feedback during simulations. Summary results are available when a simulation terminates. We introduce the ParaDiGM constructs and describe the functionality of VISA. We illustrate its utility by providing simulations of two computations under various load conditions.

**Identifier:** Diaz-Gonzalez91  
**Title:** *Language Aspects of ENVISAGER: an Object-oriented Environment for the Specification of Real-time Systems*  
**Author:** Diaz-Gonzalez, J.P.; Urban, J.E.  
**Source:** Computer Languages; vol.16, no.1; 1991; pp. 19-37  
**Date:** 1991  
**Keywords:** languages, object-oriented, real-time, formalism  
**Abstract:** Requirements engineering is the area of software engineering that deals with the study and practice of the activities related to the generation of software requirements. This paper concentrates on the concepts that are relevant to the specification of requirements for real-time systems. A specification language based on an object-oriented conceptual model is presented. Interval temporal logic, a variation of temporal logic that provides mechanisms for specifying time-varying properties of systems, is used as the underlying formalism for representing behavioral constraints on the objects. The mechanism used for the interpretation and satisfaction of the constraints is also discussed.

**Identifier:** Edmonson89  
**Title:** *DETAIL: An Approach to Task Analysis*  
**Author:** Edmonson, D.; Johnson, P.  
**Source:** Simulation and the User Interface; pp. 147-58; Brighton, UK; 18-19 May 1989  
London, UK; Taylor & Francis; 1990; 269 pp. ISBN 0 85066 803 4  
**Date:** May, 1989  
**Keywords:** methods, simulation, tools  
**Abstract:** The authors describe a tool to support task analysis, by helping users to specify steps of a task procedure. Features of the task are captured in the early stages of system development using an interactive simulation environment. The orientation of the approach is a development cycle based on rapid prototyping and simulation whereby the purpose of task analysis is to minimize the number of iterations in the prototype/evaluate cycle and so provide a commercially viable approach to interactive systems design. The approach to task analysis and modelling is intended to be timely, easy to validate and relevant; criteria which are not fully met by most existing analytic approaches. The tool has been developed using Apple Macintosh HyperCard. This is used to simulate the task in order to provide a task model which can then be refined to provide a formal input early in the development cycle.

**Identifier:** Ekambareshwar89  
**Title:** *Rapid Prototyping of Software Systems Using Prolog*  
**Author:** Ekambareshwar, S.; Downs, T.  
**Source:** Conference on Computing Systems and Information Technology 1989. Preprints of Papers; pp. 6-10; Sydney, NSW, Australia; 8-10 Aug. 1989 Sponsored by: Instn. Eng. Australia; IEEE; et al Barton, ACT, Australia; Instn. Eng. Australia; 1989; 195 pp.  
**Date:** August, 1989  
**Keywords:** languages, formalism  
**Abstract:** Rapid prototyping of software is an important method of providing customers with a means of assessing the suitability of specified requirements. This paper is concerned with a method of generating a rapid prototype for software that has been formally specified using the Z specification language. It gives an overview of Z and illustrates how example specifications can be transformed into Prolog procedures. The role of type-checking to ensure correctness of the representation is discussed.

**Identifier:** Espinosa90  
**Title:** *QUISAP: An Environment for Rapid Prototyping of Real-time Systems*  
**Author:** Espinosa, A.; Garcia-Fornes, A.; Crespo, A.; de la Puente, J.A.  
**Source:** COMPEURO '90. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering (Cat. No.90CH2867-0); pp. 502-8; Tel-Aviv, Israel; 8-10 May 1990 Sponsored by: IEEE; Inf. Processing Assoc. Israel Los Alamitos, CA, USA; IEEE Comput. Soc. Press; 1990; xiii+574 pp. ISBN 0 8186 2041 2  
**Date:** May, 1990  
**Keywords:** tools, real-time, languages, formalism, Ada  
**Abstract:** An environment for rapid prototyping and analysis of real-time systems is presented. The real-time system is specified using the language QUISAP and, from this specification, a model based on timed Petri nets for formal analysis and a prototype written in Ada for behavior analysis are built. Inappropriate handling of temporal constraints violations during prototype execution due to the Ada language can be improved with a new scheduling of tasks. Other improvements relate to the model of application objects, its definition, concurrency, and communication.

**Identifier:** Fisher87  
**Title:** *Application Software Prototyping and Fourth Generation Languages*  
**Author:** Fisher, G.E.  
**Source:** Computer Science And Technology NBS special publication; 500-148  
**Date:** May, 1987  
**Keywords:** survey

**Identifier:** Ganti90  
**Title:** *An Object-oriented Application Development Environment*  
**Author:** Ganti, M.; Goyal, P.; Nassif, R.; Podar, S.  
**Source:** COMPCON Spring '90: Thirty-Fifth IEEE Computer Society International Conference. Intellectual Leverage. Digest of Papers. (Cat. No.90CH2843-1); pp. 348-55; San Francisco, CA, USA; 26 Feb.-2 March 1990 Sponsored by: IEEE Los Alamitos, CA, USA; IEEE Comput. Soc; 1990; xvi+644 pp. ISBN 0 8186 2028 5  
**Date:** March, 1990  
**Keywords:** object-oriented, tools, reuse  
**Abstract:** The ROPE (rapid object-based prototyping environment) application development environment (ROPE-ADE) is a collection of tools that assists in the development of new software systems. One of the distinguishing features of this environment is that it assists in the reuse of both designs and code. The environment currently consists of two tools which are themselves reused in the development of other tools in the environment. One of the major aims of this environment is to facilitate rapid prototyping of applications. These prototypes are to assist in the analysis of the application and its evolution into a product. Another distinguishing feature of this environment is that it is being developed using an object-oriented design and implementation language.

**Identifier:** Gerber90  
**Title:** *Knowledge-based Software in a Realtime Alarm-handling System*  
**Author:** Gerber, S.; Baumann, R.  
**Source:** Der Elektroniker; no.10; Oct. 1990; pp. 89-94  
**Date:** October, 1990  
**Keywords:** knowledge-based, applications: process control  
**Abstract:** The authors examine the use of expert systems in analyzing alarm states in process control systems by giving particular attention to the interface between the expert system and the process itself. They illustrate this interface by describing a prototype for a PC, i.e the RTAS (realtime alarm handling system) prototype. They indicate the advantages of the method of rapid prototyping over the strictly sequential steps in the classical waterfall model for developing software and describe the iteration cycle that allows the performance, configuration and function of software to be analyzed before final creation. The features of the process PC and of the expert PC and their cooperation are described.

**Identifier:** Gimnich87  
**Title:** *Constructive Formal Specifications for Rapid Prototyping*  
**Author:** Gimnich, R.; Ebert, J.  
**Source:** Human-Computer Interaction - INTERACT '87. Proceedings of the Second IFIP Conference; pp. 1047-52; Stuttgart, West Germany; 1-4 Sept. 1987 Amsterdam, Netherlands; North-Holland; 1987; xli+1138 pp. ISBN 0 444 70304 7  
**Date:** September, 1987  
**Keywords:** formalism  
**Abstract:** The approach presented suggests a way to translate software specifications into an operational form which can be used as a prototype, for revising the requirements, and for testing purposes by relating it to the actual implementation developed later.

**Identifier:** Giordano91  
**Title:** *Rapid Development Speeds Path for Command System*  
**Author:** Giordano, F.; Wong, B.; McCollum, L.  
**Source:** Signal; vol.45, no.8; April 1991; pp. 52-6  
**Date:** April, 1991  
**Keywords:** applications: C<sup>2</sup>, real-time  
**Abstract:** The US Army European tactical command and control system (UTACCS) is a project combining rapid prototyping based on minimum, or thin, specifications with frequent operational deliveries to field users. This approach ensures that developers achieve system requirements in a timely and cost-effective manner.

**Identifier:** Gomaa81  
**Title:** *Prototyping as a Tool in the Specification of User Requirements*  
**Author:** Gomaa, H.; Scott, D.B.H.  
**Source:** IEEE CH1627-9/81/0000/0333  
**Date:** 1981  
**Keywords:** applications: process control, process  
**Abstract:** One of the major problems in developing new computer applications is specifying the user's requirements such that the Requirements Specification is correct, complete and unambiguous. Although prototyping is often considered too expensive, correcting ambiguities and misunderstandings at the specification stage is significantly cheaper than correcting a system after it has gone into production. This paper describes how a prototype was used to help specify the requirements of a computer system to manage and control a semiconductor processing facility. The cost of developing and running the prototype was less than 10% of the total software development cost.

**Identifier:** Gonzalez89  
**Title:** *Protolog: A Conceptual Schema Facility for Automated Prototype Generation*  
**Author:** Gonzalez de Rio, A.; Alpuente, M.; Cassamayor, J.C.; Pastor, M.A.; Ramirez, M.J.; Ramos, I.  
**Source:** Proceedings of the IASTED International Symposium. Applied Informatics - AI '89; pp. 43-6; Grindlewald, Switzerland; 8-10 Feb. 1989 Sponsored by: IASTED Anaheim, CA, USA; ACTA Press; 1989; 288 pp. ISBN 0 88986 117 X  
**Date:** February, 1989  
**Keywords:** tools, executable specifications, conceptual modeling  
**Abstract:** This paper presents PROTOLOG: a conceptual schema facility (CSF) for automated prototype generation. The authors introduce a new approach to conceptual modelling, which is proposed as an alternative to the operational and deductive approaches and covers the existing gap between them. The introduction of time concept is essential in the model and allows, not only to deal with historical information but also to dispose of a better expressiveness in the language. In this way, more accurate models of reality (as in the unified view of dynamic and static constraints) can be obtained. On the other hand, rapid prototyping makes the design phase easier, which allows the validation of the requirements. PROTOLOG/PG automatically translates PROTOLOG models into executable prototypes.

**Identifier:** Gregorio90  
**Title:** *A Display Rapid Prototyping and Simulation System*  
**Author:** Gregorio, D.D.; Forger, A.F.; McArdle, B.R.  
**Source:** Proceedings of the 1990 Summer Computer Simulation Conference; pp. 424-9; Calgary, Alta., Canada; 16-18 July 1990 Sponsored by: SCS San Diego, CA, USA; SCS; 1990; xix+1202 pp. ISBN 0 911801 74 X  
**Date:** July, 1990  
**Keywords:** tools, real-time, user-interface, simulation, applications: C<sup>3</sup>  
**Abstract:** The objective of the Display Rapid Prototyping and Simulation (DRPS) system is to allow its users to identify and refine both system hardware and software requirements for proposed command, control, communication and intelligence (C<sup>3</sup>I), avionics, space, armament, and other large scale defense systems by quickly prototyping user interfaces and subjecting them to the actual performance characteristics of the proposed system. DRPS does this by providing its users with the ability to: (a) define and execute a model of the target system in order to assess feasibility and performance; and (b) define within the model the operational displays of the target system and user interactions with the target system through these displays, and evaluate their operation throughout the simulation. The software is developed with VAX Ada, using a DEC version of XWindows on a DEC GPX workstation. It uses pull-down menus and graphical icons to build a system model, eliminating the need for extensive programming skills by the user. This paper examines DRPS from a user's standpoint, describing how to build a system model using the tool. It provides an overview of the tool and its capabilities. An example of a project which details the development of a model using DRPS is also provided.

**Identifier:** Gutierrez89  
**Title:** *Prototyping Techniques for Different Problem Contexts*  
**Author:** Gutierrez, O.  
**Source:** SIGCHI Bulletin; spec. issue.; May 1989; pp. 259-64 Conference on Human Factors in Computing Systems (CHI 89); Austin, TX, USA; 30 April-4 May 1989 Sponsored by: IEEE; ACM  
**Date:** May, 1989  
**Keywords:** methods, survey  
**Abstract:** Rapid prototyping and other experimental techniques are playing an increasingly important role in software development. Some common issues that concern their adoption are identifying the place in a system's life cycle where they may be appropriate, and selecting which tools to use. The author presents a model of different problem types, suggesting that a fit must be found between the nature of the problem at hand and the features associated with available techniques. Emphasis is placed on the fact that most commercial tools are suitable for only certain problem types. Some areas of further development are highlighted and implications concerning human-computer interaction discussed.

**Identifier:** **Harris87**  
**Title:** *Evaluation of Rapid Prototyping Methodology in a Human Interface*  
**Author:** Harris, J.R.; Parker, D.W.  
**Source:** Human-Computer Interaction - INTERACT '87. Proceedings of the Second IFIP Conference; pp. 1059-63; Stuttgart, West Germany; 1-4 Sept. 1987 Amsterdam, Netherlands; North-Holland; 1987; xli+1138 pp. ISBN 0 444 70304 7  
**Date:** September, 1987  
**Keywords:** applications: health care, user-interface  
**Abstract:** The authors present experiences in developing a prototype for the human interface for a database containing surveillance records for handicapped children, which will be used by clinicians and administrators in a health board authority. The user requirements were: regular assessment of a child's condition; monitoring of the child's access to, and take up of, certain special services needed; and easy compilation of figures relating to the health of the children. They describe the types of changes required by the user in the early phases of the prototype and the limitations which should be imposed on the extent of the rapid prototyping technique. They also discuss the contributions to understanding of the exact needs of the user and how iterative design methodologies achieve those needs. They highlight the benefits that speed and low investment of effort have on the design process.

**Identifier:** **Hartson91**  
**Title:** *Rapid Prototyping in Human-Computer Interface Development*  
**Author:** Hartson, H.R.; Smith, E.C.  
**Source:** Interacting with Computers; vol.3, no.1; April 1991; pp. 51-91  
**Date:** April, 1991  
**Keywords:** user-interface  
**Abstract:** Some conventional approaches to interactive system development tend to force commitment to design detail without a means for visualizing the result until it is too late to make significant changes. Rapid prototyping and iterative system refinement, especially for the human interface, allow early observation of system behavior and opportunities for refinement in response to user feedback. The role of rapid prototyping for evaluation of interface designs is set in the system development life-cycle. Advantages and pitfalls are weighed, and detailed examples are used to show the application of rapid prototyping in a real development project. Kinds of prototypes are classified according to how they can be used in the development process, and system development issues are presented. The future of rapid prototyping depends on solutions to technical problems that presently limit effectiveness of the technique in the context of present day software development environments.

**Identifier:** Hawryzkiewycz87  
**Title:** *Prototyping with the Entity-relationship Model*  
**Author:** Hawryzkiewycz, I.T.  
**Source:** Australian Computer Conference - 1987. Proceedings; pp. 332-42; Melbourne, Vic., Australia; 8-11 Sept. 1987 Watson, ACT, Australia; Australian Computer Society; 1987; xvi+904 pp.  
**Date:** September, 1987  
**Keywords:** conceptual modeling, database, methods,  
**Abstract:** One of the problems in using conceptual modelling is the transfer of analysis results rapidly into a working system. In many cases the conceptual model is transferred to database by a manual process and as a result the model structure is either partially ignored or incorrectly transferred into an implementation. Manual conversion also does not meet the requirement of many end-user systems, which are based on prototyping or evolutionary design rather than the linear development cycle. Again this requirement needs a rapid transfer of any data model to the implementation and calls for support of changes to the system. These problems can be overcome with software support for conceptual modelling and conversion from the model to a database definition. Furthermore prototyping can be supported by special conversions that generate screens and reports of primitive operations such as adding or deleting entities and relationships and constraint checks. The paper describes the development of a system for rapid prototyping based on these ideas. It describes an E-R modelling tool that assists end users to develop correct E-R models and converts these models directly into a database definition for an application generator. The paper then outlines further requirements that automated tools must satisfy to support prototyping. These include screen and report generation and support for change at the conceptual model level.

**Identifier:** Helsler88  
**Title:** *Integrating the Role of Rapid Prototyping and Requirements Specification Using the Object-oriented Paradigm*  
**Author:** Heisler, K.G.; Tsai, W.T.  
**Source:** Technical report. University of Minnesota. Institute of Technology. Computer Science Dept.; TR-88-65  
**Date:** 1988  
**Keywords:** object-oriented

**Identifier:** Hekmatpour90  
**Title:** *AI Techniques and Object-oriented Technology for VLSI Design-space Representation, Optimization and Management*  
**Author:** Hekmatpour, A.; Chau, P.  
**Source:** Proceedings of the SPIE - The International Society for Optical Engineering; vol.1293, pt.1; 1990; pp. 85-94 Applications of Artificial Intelligence VIII; Orlando, FL, USA; 17-19 April 1990 Sponsored by: SPIE  
**Date:** April, 1990  
**Keywords:** object-oriented, knowledge-based, tools  
**Abstract:** The VLSI design process consists of many highly specialized tasks. Algorithmic, computationally-intensive and mundane tasks which used to be performed manually, have been automated by traditional VLSI CAD tools. These tools have automated many aspects of VLSI design synthesis, analysis, optimization and verification. However, the successful and efficient utilization of these tools has proved to be very knowledge intensive, requiring the interaction and guidance of domain experts. In recent years, many expert systems for VLSI design have been reported, but these stand-alone expert systems have to be integrated with traditional CAD tools to be able to provide automated decision-making, judgment-based opportunity ranking and tighter data source integration. Furthermore, an integrated and distributed knowledge-base is essential for rapid prototyping of expert design assistants, design techniques and optimization heuristics. The authors report an experimental VLSI design environment based on a central object-oriented model-base. The object-oriented design kernel is capable of accommodating traditional CAD tools as well as knowledge-based tools.

**Identifier:** Henskes87  
**Title:** *Rapid Prototyping of Man-machine Interfaces for Telecommunications Equipment Using Interactive Animated Computer Graphics*  
**Author:** Henskes, D.T.; Tolmie, J.C.  
**Source:** Human-Computer Interaction - INTERACT '87. Proceedings of the Second IFIP Conference; pp. 1053-8; Stuttgart, West Germany; 1-4 Sept. 1987 Amsterdam, Netherlands; North-Holland; 1987; xli+1138 pp. ISBN 0 444 70304 7  
**Date:** September, 1987  
**Keywords:** user-interface, graphical specification, simulation  
**Abstract:** The concept of rapid prototyping can be extended from software development support to simulation of man-machine interfaces. This approach will help meeting the challenge imposed on telecommunications engineering by the evolution of an European broadband network system with its subsequent need for highly acceptable user services. Animated computer graphics is a cost effective way for introducing simulation into the earliest possible phase of the design cycle

**Identifier:** Henskes90  
**Title:** *Prototyping and Visualisation in Interface Design*  
**Author:** Henskes, D.T.; Tolmie, J.C.  
**Source:** *Electrical Communication*; vol.64, no.4; 1990; pp. 321-6  
**Date:** 1990  
**Keywords:** user-interface, tools, simulation  
**Abstract:** To meet the needs of software engineering in telecommunications, Alcatel SEL has created a design and simulation toolkit. The system is built on the X11 window system standard and the OSF/Motif widget set, thereby providing portability across a wide range of hardware platforms. The toolkit separates the components of the software system, allowing the user interface to be constructed independently of the application. This strict isolation of the user interface allows easy integration of the tools into existing software engineering environments. Emphasis is placed on high quality visualisation and realism. A dialogue component enables the designer to stimulate the application, making the toolkit ideal for rapid prototyping. Object-oriented techniques have been used, permitting the interface objects to be used by other applications. The toolkit is being used to prototype a wideband cross-connect system Multipart, which is a modification of the Alcatel CX4111.

**Identifier:** Hughes89  
**Title:** *The Emergence of Rapid Prototyping as a Real-time Software Development Tool*  
**Author:** Hughes, T.S.; Cooling, J.E.  
**Source:** IEE Colloquium on 'Specification of Complex Systems' (Digest No.145); pp. 2/1-3; London, UK; 30 Nov. 1989 Sponsored by: IEE London, UK; IEE; 1989; 29 pp.  
**Date:** November, 1989  
**Keywords:** real-time, methods  
**Abstract:** Recent years have seen an explosion in the demand for high quality software. The complexity and size of these projects has also increased. Software developers have been forced to recognize the shortcomings of conventional methods for producing software. As new methods for controlling and managing production have emerged, researchers have emphasized the importance of the system specification. The task of constructing the system specification involves a great deal of communication between the developers and the clients. The use of specialized technical language by both parties makes precise exchange of ideas difficult. There is thus a need for both parties to be able to explore a commonly agreed definition of the problem. This requires a method of presenting the problem definition which is easily understood by both parties. The use of rapid prototyping is one approach to solving this problem and the authors discuss its use as a software development tool.

**Identifier:** IEE89  
**Title:** *IEE Colloquium on 'Specification of Complex Systems' (Digest No. 145*  
**Source:** London, UK; 30 Nov. 1989 Sponsored by: IEE London, UK; IEE; 1989; 29 pp.  
**Date:** 1989  
**Keywords:** structured analysis, real-time  
**Abstract:** The following topics were dealt with: rapid prototyping as a real-time software development tool; specification and procurement of complex systems; structured analysis at VSEL; mathematical precision and user understanding; project feasibility; and specification quality assurance

**Identifier:** Jain89  
**Title:** *Software Quality via Rapid Prototyping*  
**Author:** Jain, A.K.; Tink, P.D.  
**Source:** GLOBECOM '89. IEEE Global Telecommunications Conference and Exhibition. Communications Technology for the 1990s and Beyond (Cat. No.89CH2682-3); pp. 642-6 vol.1; Dallas, TX, USA; 27-30 Nov. 1989 Sponsored by: IEEE New York, NY, USA; IEEE; 1989; 3 vol. xxxii+1975 pp.  
**Date:** November, 1989  
**Keywords:** process, applications: networks  
**Abstract:** Rapid prototyping is described as a software development process which provides improved software quality and increases software productivity, in addition to providing other benefits such as reduced costs, reduced risk, shorter schedules, and customer satisfaction. This process does not usually produce throwaway prototypes; rather, prototyping methodology is employed without using any specific rapid prototyping tools to produce field-grade products. The approach emphasizes selection of smaller teams working closely with the customers. Using rapid prototyping, many successful software systems that perform planning, provisioning, monitoring, and maintenance operations support for various AT&T network services have been developed.

**Identifier:** Jones90  
**Title:** *HyperCard- The Legend: Summary*  
**Author:** Jones, M.L.R.  
**Source:** IEE Colloquium on 'Software Tools for Interface Design' (Digest No.146); pp. 4/1-4; London, UK; 8 Nov. 1990 Sponsored by: IEE London, UK; IEE; 1990; 78 pp.  
**Date:** November, 1990  
**Keywords:** tools  
**Abstract:** The paper is intended as an essentially practical account of the strengths and weaknesses of HyperCard as a rapid prototyper of product interfaces. It is assumed that the reader is familiar with the basic components of HyperCard: stacks, cards, backgrounds, fields, buttons and scripts.

**Identifier:** Koch88  
**Title:** *Methodical and Management Experiences from an Extensive Software Project*  
**Author:** Koch, W.H.  
**Source:** Experience with the Management of Software Projects 1988. Proceedings of the 2nd IFAC/IFIP Workshop; pp. 31-6; Sarajevo, Yugoslavia; 27-29 Sept. 1988 Sponsored by: IFAC; IFIP Oxford, UK; Pergamon; 1990; viii+95 pp. ISBN 0 08 036928 6  
**Date:** September, 1988  
**Keywords:** methods, survey  
**Abstract:** A project involving the software redevelopment for durable investment goods is discussed. The major pre-design, design and implementation decisions are reflected. Important topics are the use of rapid prototyping as a design aid and the use of symbolic debuggers in a multitasking environment. Finally the most important methods used are reviewed and suggestions for future developments are made.

**Identifier:** Konrad  
**Title:** *Functional Prototyping with Proto*  
**Author:** Konrad, M.I D.; Welch, T. A.  
**Source:** Book: *Sample CASE Tools and Perspectives in CASE*, Chapter 12  
**Keywords:** languages, reuse, tools, methods, executable specifications  
**Abstract:** Rapid prototyping is pursued as a means to develop and validate functional specifications prior to extensive code development in a large software system. Four critical components of a functional prototyping capability are: 1) a language for specifying the functions being examined, 2) a library of reusable software modules to expedite specification, 3) a set of interactive tools for constructing and analyzing the specification, and 4) a methodology that guides the analyst in construction, analysis, and validation of the functional specification. The authors illustrate how they have realized these four components in Proto, a functional prototyping capability they have developed. In Proto, one creates a specification of the functionality to be examined, augments it to be an executable functional prototype, and validates it via demonstration of system capabilities to potential end-users.

**Identifier:** Kordon90  
**Title:** *Rapid Ada Prototyping: Principles and Example of a Complex Application*  
**Author:** Kordon, F.; Estraillier, P.; Card, R.  
**Source:** Ninth Annual International Phoenix Conference on Computers and Communications (Cat. No.90CH2799-5); pp. 453-60; Scottsdale, AZ, USA; 21-23 March 1990 Sponsored by: IEEE; Arizona State Univ.; Univ. Arizona Los Alamitos, CA, USA; IEEE Comput. Soc. Press; 1990; xxii+910 pp. ISBN 0 8186 2030 7  
**Date:** March, 1990  
**Keywords:** applications: telephony, formalism, real-time  
**Abstract:** The automatic prototyping methodology presented is derived from Petri net theory and has been developed for the PNTAGADA project (Petri Net Translation, Analysis, and Generation of Ada code). Colored Petri nets allow concise modeling and verification of distributed systems. Their quantitative analysis provides invariants which are of particular interest for rapid prototyping of parallel applications. Management of a phone conversation, or a complex application, is used as an example to demonstrate the methodology of the Ada code generator. A description of subscriber behavior and services available at a private automatic branch exchange (PABX) is presented, along with a qualitative analysis of the model. A listing abstract of the task associated with a given subscriber is provided. Execution of the code generated for that model presents the process sequence involved in management of different subscribers.

**Identifier:** Kozubal90  
**Title:** *Run-time Environment and Application Tools for the Ground Test Accelerator Control System*  
**Author:** Kozubal, A.J.; Kerstiens, D.M.; Hill, J.O.; Dalesio, L.R.  
**Source:** Nuclear Instruments & Methods in Physics Research, Section A (Accelerators, Spectrometers, Detectors and Associated Equipment); vol.A293, no.1-2; 1 Aug. 1990; pp. 288-91 International Conference on Accelerator and Large Experimental Physics Control Systems; Vancouver, BC, Canada; 30 Oct.-3 Nov. 1989 Sponsored by: TRIUMF; Eur. Phys. Soc.; IEEE  
**Date:** August, 1990  
**Keywords:** applications: control systems, distributed processing  
**Abstract:** The control system for the ground test accelerator (GTA) at Los Alamos provides capabilities and tools that considerably reduce the amount of programming required to perform many applications. These qualities have proved to be valuable on early GTA experiments, where rapid prototyping has paid off. For instance, the initial controls for a 1 MW RF power supply provided supervisory control with no application-dependent programming. These same qualities will enable the authors to automate the start-up, operation and shutdown of the GTA. The run-time environment makes effective use of the distributed, nonhierarchical control-system architecture by providing a standard interface to the distributed data base. This paper gives an overview of the run-time software environment and the tools that simplify building the run-time data base, the operator interface screens, and application-specific control operations-sequential and continuous.

**Identifier:** Kreutzer90a  
**Title:** *The Modeller's Assistant- A First Step Towards Integration of Knowledge Bases and Modelling Systems*  
**Author:** Kreutzer, W.  
**Source:** Proceedings of the 1990 Summer Computer Simulation Conference; pp. 874-9; Calgary, Alta., Canada; 16-18 July 1990 Sponsored by: SCS San Diego, CA, USA; SCS; 1990; xix+1202 pp. ISBN 0 911801 74 X  
**Date:** July, 1990  
**Keywords:** tools, object-oriented, graphical specification, knowledge-based, simulation  
**Abstract:** The Modeller's Workbench project explores the benefits of applying object orientation, a methodology of layered design, exploratory programming, graphical interaction, and knowledge-based software architectures to the domain of simulation development and execution. One of its facets provides rapid prototyping of graphically animated queueing scenarios using a 'desktop-style' modelling environment written in Smalltalk, and the 'Modeller's Assistant' is the name of an expert system supporting this framework. The simulation method and AI-style knowledge-based systems are characterized and contrasted, leading to the conclusion that all expert systems are essentially 'model-based'. The Modeller's Assistant's architecture and a simple example of the way in which it is intended to be used from the core of the paper. The author concludes with a discussion of further research in this area, and some comments on promise and problems of modelling environments characterized by large numbers of interacting knowledge sources.

**Identifier:** Kreutzer90b  
**Title:** *Tiny Tim-a Smalltalk Toolbox for Rapid Prototyping and Animation of Models*  
**Author:** Kreutzer, W.  
**Source:** *Journal of Object-Oriented Programming*; vol.2, no.5; Jan.-Feb. 1990; pp. 27-36  
**Date:** February, 1990  
**Keywords:** tools, graphical specification, simulation  
**Abstract:** The motivation and justification for developing a modeller's workbench are discussed. The author goes on to describe Tiny Tim, a Smalltalk toolbox for graphical model design and animation. The advantages of using Smalltalk, with its object-oriented programming approach, are considered. To demonstrate the desktop-style modelling interface of Tiny Tim, a Monte Carlo simulation is conducted. Future developments in simulation systems for desktop development environments are also discussed.

**Identifier:** Krista89  
**Title:** *A Computer Aided Prototyping Methodology*  
**Author:** Krista, R.; Rozman, I.  
**Source:** SIGSOFT Software Engineering Notes; vol.14, no.6; Oct. 1989; pp. 68-72  
**Date:** October, 1989  
**Keywords:** methods, graphical specification, structured analysis  
**Abstract:** A methodology for rapid prototyping is described. Modified data flow diagrams are used as a graphical tool and the prototyping system description language is developed. The advantages of the methodology and of the prototyping tool, which is being build, are as follows: a possibility to describe a system in an understandable graphical way without global data, a possibility to describe the system by parallel activities and a possibility to verify the whole system or only a part of it. A simple and understandable specification of the new system is enabled by decomposition.

**Identifier:** Lea90  
**Title:** *Rapid Prototyping from Structured Analysis: Executable Specification Approach*  
**Author:** Lea, R.J.; Chung, C.G.  
**Source:** *Information and Software Technology*; vol.32, no.9; Nov. 1990; pp. 589-97  
**Date:** November, 1990  
**Keywords:** executable specifications, graphical specification, structured analysis  
**Abstract:** A rapid prototyping approach is proposed, by which an executable prototype can be quickly constructed from the result of structured analysis (SA). Thus user feedback can be obtained earlier in requirement analysis. To accord with SA, two different specification schemes specify the operation logic of system service functions and the characteristics of data objects. The prototyping procedure and the environment for supporting prototype execution are also presented.

**Identifier:** Luckey90  
**Title:** *Rapid Prototyping in Ada in the Rational Environment Emphasizing Software Reuse*  
**Author:** Luckey, P.H.; DuPont, F.G.  
**Source:** Proceedings of the Seventh Washington Ada Symposium; pp. 71-5; McLean, VA, USA; 25-28 June 1990 Sponsored by: ACM; NASA; Fed. Aviation Adm.; et al New York, NY, USA; ACM; 1990; 341 pp.  
**Date:** June, 1990  
**Keywords:** Ada, reuse, database  
**Abstract:** A recent experience at IBM/FSD Owego demonstrates how prototyping in Ada is enhanced via the incorporation of software reuse technologies in an integrated development environment. In response to a recent new business proposal at Owego, a user interface for a database application was prototyped. The purpose of the prototyping exercise was three fold: (1) to aid in the size estimation of a program to be developed; (2) to confirm the viability of developing the program in Ada; and (3) to demonstrate the productivity possible when developing with reuse in mind in the Rational environment. The results of the exercise were that the purpose was accomplished and an object-oriented prototyping process was developed.

**Identifier:** Luqi86  
**Title:** *Rapid Prototyping for Large Software System Design*  
**Author:** Luqi  
**Source:** Dissertation, University of Minnesota; UMI Dissertation Service  
**Date:** 1986  
**Keywords:** methods, real-time, graphical specification

**Identifier:** Luqi87  
**Title:** *Research Aspects of Rapid Prototyping*  
**Author:** Luqi.  
**Source:** Defense Technical Information Center, AD-A179 007; NPS52-87-006  
**Date:** 1987  
**Keywords:** methods, real-time, graphical specification

**Identifier:** Luqi89  
**Title:** *Software Evolution through Rapid Prototyping*  
**Author:** Luqi  
**Source:** IEEE Computer  
**Date:** May, 1989  
**Keywords:** tools, methods, process

**Identifier:** Luq190a  
**Title:** *Graphical Support for Reducing Information Overload in Rapid Prototyping*  
**Author:** Luqi; Barnes, P.D.; Zyda, M.  
**Source:** Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences; pp. 514-22 vol.2; Kailua-Kona, HI, USA; 2-5 Jan. 1990  
Sponsored by: IEEE; Univ. Hawaii; PRIISM; ACM Los Alamitos, CA, USA; IEEE Comput. Soc. Press; 1990; 4 vol. (x+449+xii+575+xii+673+xi+515) pp.  
**Date:** January, 1990  
**Keywords:** graphical specification tools  
**Abstract:** The authors discuss the capability of graphical representations to ease the prototyping process and reduce the problem of information overload. The application of information-hiding and multiple views, coupled with ensuring consistency and automatic programming, can improve user productivity. The development of a graphical editor for performing hierarchical decomposition of composite PSDL (prototype system description language) operators for CAPS (computer aided prototyping system) is also discussed. Research on the graphical editor, as it relates to PSDL, indicates that a prototype design can be developed with much greater ease using the graphical editor than with only the syntax-directed editor. The graphical editor will also enhance prototype modification, presentation, and documentation.

**Identifier:** Luq190b  
**Title:** *Graphical Tool for Computer-aided Prototyping*  
**Author:** Luqi; Barnes, P.D.; Zyda, M.  
**Source:** Information and Software Technology; vol. 32, no. 3; pp. 199-206  
**Date:** April, 1990  
**Keywords:** tools, graphical specification, real-time  
**Abstract:** The basic problem in rapid prototyping of software is information overload. Graphic interfaces can help by providing multiple views, where each view is limited to providing information relevant to a particular task or problem. The graphical editor under development for the computer aided prototyping system (CAPS) proposes a dataflow-diagram-based model with multiple views and automatic program generation to manage the quantity of information necessary to prototype large, real-time systems.

**Identifier:** Luq191  
**Title:** *CAPS as a Requirements Engineering Tool*  
**Author:** Luqi; Steigerwald, R.; Hughes, G.; Naveda, F.; Berzins, V.  
**Source:** Proceedings, Requirements Engineering and Analysis Workshop, SEI  
**Date:** 1991  
**Keywords:** process, tools, executable specifications  
**Abstract:** The process of determining user requirements for software systems is often plagued with uncertainty, ambiguity, and inconsistency. Rapid prototyping offers an iterative approach to requirements engineering to alleviate the problems inherent in the process. CAPS (the Computer Aided Prototyping System) has been built to help software engineers rapidly construct software prototypes of proposed software systems. We describe how CAPS as a prototyping tool helps firm up software requirements through iterative negotiations between customers and designers via examination of executable prototypes.

**Identifier:** Luq192  
**Title:** *Computer-Aided Prototyping for a Command-and-Control System Using CAPS*  
**Author:** Luqi  
**Source:** IEEE Software  
**Date:** January, 1992  
**Keywords:** process, tools, executable specifications

**Identifier:** Madison89  
**Title:** *Rapid Prototyping for Healthcare Applications*  
**Author:** Madison, D.E.  
**Source:** Computers in Healthcare; vol.10, no.11; Nov. 1989; pp. 35-6, 38  
**Date:** November, 1989  
**Keywords:** applications: health care, user-interface, knowledge-based, logic programming  
**Abstract:** One way automation can increase efficiency is through rapid prototyping, particularly of user interfaces. AI and logic programming offer good user interface tools and a suitable prototyping environment.

**Identifier:** Matsumoto  
**Title:** *Japanese Perspectives in Software Engineering*  
**Author:** Matsumoto, Y.; Ohno, Y.  
**Source:** Book  
**Keywords:** survey, collection

**Identifier:** McEnery90  
**Title:** *Pantheon: Rapid Prototyping of Natural Language Interfaces to Large Databases*  
**Author:** McEnery, A.M.; Oakes, M.P.; Reid, D.  
**Source:** Proceedings of the 12th BCS IRSG Research Colloquium on Information Retrieval; pp. 135-51; Huddersfield, UK; 3-5 April 1990 Huddersfield, UK; Polytech. Huddersfield; 1990; 160 pp.  
**Date:** April, 1990  
**Keywords:** database, tools  
**Abstract:** Pantheon is a software package of natural language processing units that have been developed at Lancaster and Liverpool over the past two years. The aim of Pantheon has been to provide a natural language processing facility to applications requiring it by providing specific modules that can be tailored to applications. Pantheon has been used to convert limited natural language input into legal search terms for a large pharmaceutical database, RINGDOC, which is published by Derwent Publications Ltd. In parallel to the development of the natural language interface for the system, a suite of menu generating programs, Greek, were developed to carry out a parallel study into the feasibility of menu based approaches. The Greek package was concerned with generating menus to a specific application area rapidly, and enabling search intermediaries to encode their knowledge of a domain in a structured format. In interfacing the packages to the database a critical evaluation was undertaken between the merits and demerits of both natural language and menu driven interactions. The paper shows how the use of the Pantheon and Greek suites of programs facilitated rapid prototyping of user interfaces. It considers some of the realities of linguistics and interfacing, and on this basis compares the interfaces in terms of user friendliness and functionality.

**Identifier:** McInroy89  
**Title:** *Rapid Prototyping Capabilities in the Expert Requirements Expression and System Synthesis (EXPRESS) Environment*  
**Author:** McInroy, J.W.  
**Source:** Proceedings of the SPIE - The International Society for Optical Engineering; vol.1095, pt.2; 1989; pp. 1020-30 Applications of Artificial Intelligence VII; Orlando, FL, USA; 28-30 March 1989 Sponsored by: SPIE  
**Date:** March, 1989  
**Keywords:** tools, knowledge-based, executable specifications  
**Abstract:** The Expert Requirements Expression and System Synthesis (EXPRESS) environment is being developed at the Lockheed Software Technology Center in Palo Alto, California. EXPRESS provides rapid prototyping and will support full-scale engineering development (FSED) via integrated, knowledge-based, executable specifications and related capabilities. That is, EXPRESS provides automatic programming via two key technologies: (1) executable specifications, written in very high-level languages (VHLLs) and (2) knowledge base technology. EXPRESS provides four integrated, very high-level specification languages. EXPRESS is based on the Refine language and knowledge base management system. Specifications written in the four specification languages are translated into Refine language constructs and knowledge base constructs, through which the specifications communicate. The initially resulting code is not optimized, but it serves for a rapid prototype. Capabilities both for humans to make design decisions and for EXPRESS to perform automatic optimization are being explored. EXPRESS runs on networks of Symbolics workstations, at both system specification time and at run time.

**Identifier:** **Milovanovic90**  
**Title:** *Experience with the Management of Software Projects 1988*  
**Author:** Milovanovic, R.; Elzer, P.  
**Source:** Proceedings of the 2nd IFAC/IFIP Workshop Sarajevo, Yugoslavia; 27-29 Sept. 1988 Sponsored by: IFAC; IFIP Oxford, UK; Pergamon; 1990; viii+95 pp. ISBN 0 08 036928 6  
**Date:** 1990  
**Keywords:** survey  
**Abstract:** The following topics were dealt with: software project support tools and environments; project team management; rapid prototyping; large project management; software metrics experiences; software quality; safety critical software; software productivity; and CASE tools.

**Identifier:** **Minkowitz89**  
**Title:** *Software Architecture Modelling*  
**Author:** Minkowitz, C.  
**Source:** Software Engineering for Large Software Systems; pp. 325-44; Bristol, UK; 26-29 Sept. 1989 Barking, UK; Elsevier Applied Science Publishers; 1990; x+373 pp. ISBN 1 85166 504 8  
**Date:** September, 1989  
**Keywords:** formalism, object-oriented, methods, tools  
**Abstract:** Techniques such as formal methods and object-oriented design allow software engineers to describe the structure and design of a system at a high level of abstraction. They free software engineers from concerns about implementation details so that the engineers can concentrate on the gross organization of the data structures and algorithms that constitute the system. This kind of software architecture modelling enables software engineers to explore the design space of a system and to clear up conceptual errors and misunderstandings about a system's basic structure. The me too method makes use of three techniques-formal specification, functional programming and rapid prototyping-to model software architecture. This paper discusses the use of me too to model the architecture of part of a large software system that is being developed for the Esprit IMSE (Integrated Modelling Support Environment) project.

**Identifier:** Nugent88  
**Title:** *A Distributed Interactive Scenario Generator for Command, Control And Communications*  
**Author:** Nugent, E.R.; Moody, S.A.  
**Source:** Proceedings of the 1988 Summer Computer Simulation Conference; pp. 536-41; Seattle, WA, USA; 25-28 July 1988 Sponsored by: SCS San Diego, CA, USA; SCS; 1988; xxxiv+960 pp. ISBN 0 911801 38 3  
**Date:** July, 1988  
**Keywords:** user-interface, simulation, applications: C<sup>3</sup>I  
**Abstract:** The Interactive Scenario Generator (ISG) is a simulation-based software tool designed to provide a dynamic environment model for RAPID (Rapid Prototyping of Interface Design). RAPID is a collection of distributed hardware and software components executing on a local area network. It is used by computer-literate domain experts to model command, control, communication and intelligence (C<sup>3</sup>I) workstation interface concepts. The ISG emulates message traffic arriving at the workstation as defined by the operational model that is employed. It has the capability to respond to workstation actions that affect the course of the scenario. This provides a live environment for testing workstation interface concepts. Scenarios are modeled using a high-level general-purpose simulation language. Interprocess communication allows the simulation to provide real-time status messages to the prototype workstation. This paper presents an overview of the RAPID environment and provides a detailed discussion of the ISG implementation.

**Identifier:** O'Neil89  
**Title:** *Rapid Prototyping of Formal Specifications Using Miranda*  
**Author:** O'Neil, G.  
**Source:** Nat. Phys. Lab., Teddington, UK; NPL DITC 150/89; Nov. 1989; 30 pp.  
**Date:** November, 1989  
**Keywords:** formalism, executable specifications, database  
**Abstract:** A discussion is given on the use of the functional programming language Miranda, in the early stages of the software lifecycle, to produce executable versions of formal specifications. Two examples are given; the first showing a rapid-prototyping approach in which the user requirements and the formal specification for a small database system are developed together by means of a Miranda prototype and the second illustrating the animation of a VDM specification using Miranda.

**Identifier:** Ortner88  
**Title:** *Dictionary-supported Prototyping of Database Applications*  
**Author:** Ortner, E.; Rohrle, J.  
**Source:** Experience with the Management of Software Projects 1988. Proceedings of the 2nd IFAC/IFIP Workshop; pp. 21-5; Sarajevo, Yugoslavia; 27-29 Sept. 1988 Sponsored by: IFAC; IFIP Oxford, UK; Pergamon; 1990; viii+95 pp. ISBN 0 08 036928 6  
**Date:** September, 1988  
**Keywords:** database, methods  
**Abstract:** *Rapid Prototyping as a software development method should not be implemented without regard to existing applications. Since prototypes are produced with the aid of special tools, the question arises, how relevant components of existing applications can be integrated into prototype development and how the results of the prototyping process can be integrated into existing applications. One possible way of achieving this goal is communication between the enterprise data dictionary for existing applications and the data dictionary of the prototyping environment on the basis of a common documentation structure. This process of exchanging metadata between the two dictionary systems must be guaranteed by suitable tools and consistency enforcement routines.*

**Identifier:** Overmyer90  
**Title:** *The Impact of DoD-Std-2167A on Iterative Design Methodologies: Help or Hinder?*  
**Author:** Overmyer, S.P.  
**Source:** SIGSOFT Software Engineering Notes; vol.15, no.5; Oct. 1990; pp. 50-  
**Date:** October, 1990  
**Keywords:** process  
**Abstract:** Many experts in software engineering agree that the emerging iterative requirements engineering software engineering and software design methodologies present excellent ways to identify and validate user requirements. These methodologies often include innovative techniques for elicitation and validation of user requirements including various forms of human engineering analysis, rapid prototyping, and knowledge acquisition tasks. The paper addresses the compatibility of these techniques with DoD-Std-2167A. Assessment is made regarding the compatibility of the standard with innovative requirements techniques, and how and where these techniques may be inserted into the life cycle.

**Identifier:** Powers89  
**Title:** *Ensemble: A Graphical User Interface Development System for the Design and Use of Interactive Toolkits*  
**Author:** Powers, M.K.  
**Source:** UIST. Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology; pp. 168-79; Williamsburg, VA, USA; 13-15 Nov. 1989  
Sponsored by: ACM New York, NY, USA; ACM; 1989; vii+179 pp. ISBN 0 89791 335 3  
**Date:** November, 1989  
**Keywords:** user-interface, tools  
**Abstract:** User interface development systems (UIDS), as opposed to user interface management systems or UI toolkits focus on supporting the design and implementation of the user interface. The paper describes Ensemble, an experimental UIDS that begins to explore the electronic creation of interaction techniques as well as the corresponding design processes. Issues related to the impact on the components of the development system are discussed. Finally, problems with the current implementation and future directions are presented.

**Identifier:** Purtillo91  
**Title:** *A Methodology for Prototyping-in-the-Large*  
**Author:** Purtillo, J.; Larson, A.; Clark, J.  
**Source:** Proceedings, IEEE 13th International Conference on Software Engineering  
**Date:** 1991  
**Keywords:** methods, languages, tools  
**Abstract:** Just as programming-in-the-small entails fundamentally different activities from programming-in-the-large, so is prototyping necessarily different when performed within very large scale applications. This paper defines prototyping as an experimental activity intended to reduce risk of failure in a software product. In this context, we explore the effect of scale in prototyping, then describe a methodology for prototyping a large application. Next we describe a system being developed to evaluate this methodology, featuring a pair of languages (Promo and Moblog) to serve both large-scale and component-level prototyping needs. We conclude with a presentation of how our methodology would be applied to a sample problem, a fault-prediction subsystem within the Space Station Freedom Project.

**Identifier:** Rizman90a  
**Title:** *Using Data-flow Description Supported by the Rapid Prototyping Tool for Specifying and Developing of Knowledge-based Systems*  
**Author:** Rizman, K.; Rozman, I.  
**Source:** SEKE '90. Proceedings. Software Engineering and Knowledge Engineering. 2nd International Conference; pp. 58-63; Skokie, IL, USA; 21-23 June 1990 Sponsored by: Knowledge Syst. Inst.; Inst. Inf. Ind.; S.W.I.F.T.; Univ. Pittsburgh Skokie, IL, USA; Knowledge Syst. Inst; 1990; iii+277 pp.  
**Date:** June, 1990  
**Keywords:** graphical specification knowledge-based, tools, structured analysis  
**Abstract:** A rapid prototyping tool-RPT for the development of knowledge-based systems is described. The RPT enables rapid prototyping of knowledge-based systems associated with the more traditional software development. The fact that describing knowledge bases directly corresponds to the description by means of data flow diagrams/data dictionary (DFD/DD) used for program specifications is investigated and applied. Data flow diagrams give a nonprocedural program description used in software engineering and a graphical representation of knowledge bases used in knowledge engineering. The development by the help of the RPT is focussed on how outputs will be computed from inputs, instead of what transformations (the structure of knowledge base's rules) are required. The RPT provides a natural hierarchical way to describe and document knowledge-based systems. It enables the execution of knowledge-based prototypes and the testing of the correctness of their structure and contents.

**Identifier:** Rizman90b  
**Title:** *A Rapid Prototyping Approach to Software Development and our Tool for Development of RPT Prototypes*  
**Author:** Rizman, K.; Rozman, I.; Verber, D.  
**Source:** Elektrotehniski Vestnik; vol.57, no.4; Aug.-Oct. 1990; pp. 294-8  
**Date:** October, 1990  
**Keywords:** methods, tools  
**Abstract:** The paper presents research in automating the software development process carried out at the Laboratory for Informatics at the Faculty of Technical Sciences in Maribor. The rapid prototyping approach to software development is described together with the rapid prototyping tool-RPT, which is being designed.

**Identifier:** Royce89  
**Title:** *Reliable, Reusable Ada Components for Constructing Large, Distributed Multi-task Networks: Network Architecture Services (NAS)*  
**Author:** Royce, W.  
**Source:** Proceedings. TRI-Ada '89; pp. 500-16; Pittsburgh, PA, USA; 23-26 Oct. 1989 New York, NY, USA; ACM; 1989; xxvi+670 pp. ISBN 0 89791 329 9  
**Date:** October 1989  
**Keywords:** Ada, reuse, distributed processing, applications: networks  
**Abstract:** The key concepts of TRW's Reusable Message Based Design Software (Network Architecture Services-NAS) which has proven to be key to the CCPDS-R project's progress to date, are presented. The NAS software and supporting tools have provided the CCPDS-R project team with reliable, powerful building blocks that have been integrated into extensive demonstrations to validate the critical design approaches. The CCPDS-R PDR demonstration consisted of 130 Ada tasks interconnected via 450 different task to task interfaces, executing in a network of 3 VAX nodes. The advantages of NAS usage are twofold: value added operational software through reuse of mission independent, performance tunable components which support open architectures; and overall project productivity enhancement as a result of NAS support for rapid prototyping, runtime instrumentation toolsuite, and encapsulation of the difficult capabilities required in any distributed real-time system into a standard set of building blocks with simple applications interfaces. The author describes the message based design techniques which led to the development of NAS, the capabilities and components inherent in the NAS product and the CCPDS-R experience in using NAS in a stringent real time command and control environment.

**Identifier:** Rzepka86  
**Title:** *A Prototyping Tool to Assist in Requirements Engineering*  
**Author:** Rzepka, W. E.; Daley, P.C.  
**Source:** Proceedings of the Nineteenth Annual Hawaii International Conference on Systems Sciences  
**Date:** 1986  
**Keywords:** tools, real-time, reuse, user-interface, methods  
**Abstract:** Rome Air Development Center is currently developing a Requirements Engineering Testbed whose goal is to support the research and development of methods and tools for the purpose of applying and evaluating them in the development of requirements for Air Force embedded computer systems. The testbed will incorporate several techniques. This paper describes one of its components-- a prototyping tool to assist in requirements engineering. This tool's objective is to help requirements engineers answer questions about key facets of embedded computer systems early in their conceptual development. The questions concern aspects of the system's operator/analyst interface, subsystem communications, system data flow, operator work flow, data base management strategy and system performance. The tool utilizes existing software packages which can be easily parameterized to quickly fabricate and model relevant aspects of embedded computer systems. The result is a combination of interface, functional and performance prototypes which assist the requirements engineer in improving his understanding of a system's requirements. The prototypes are "throwaways" in the sense that they are not intended to become part of the eventual system development, but are retained for reuse in examining the requirements of other systems. This paper describes the conceptual requirements and design of the tool, a methodology for using it, and the results, of building a prototype to demonstrate its feasibility and validate its user interface requirements.

**Identifier:** SEKE90  
**Title:** *SEKE '90. Proceedings. Software Engineering and Knowledge Engineering. 2nd International Conference*  
**Source:** SEKE '90, Skokie, IL, USA; 21-23 June 1990 Sponsored by: Knowledge Syst. Inst.; Inst. Inf. Ind.; S.W.I.F.T.; Univ. Pittsburgh Skokie, IL, USA; Knowledge Syst. Inst; 1990; iii+277 pp.  
**Date:** 1990  
**Keywords:** collection  
**Abstract:** The following topics were dealt with: knowledge-based software engineering methodologies; object-oriented systems; rapid prototyping; AI research; knowledge-based systems; software engineering of expert systems; logic programming and knowledge representation; database systems; software design; and graphics and graph models

**Identifier:** **Shirota89**  
**Title:** *Speciication and Automatic Generation of Intelligent Graphical Interfaces*  
**Author:** Shirota, Y.; Kunii, T.L.  
**Source:** Technical report. University of Tokyo. Faculty of Science. Dept. of Information Science; 89-026  
**Date:** 1989  
**Keywords:** user-interface, methods, tools  
**Abstract:** This paper outlines a new type of visual interface called Enhanced Menu-Based Software (EMBS), and describes a visual specification and automatic generation method for such software. In EMBS, data management facilities of spreadsheets, and CAD facilities are successfully integrated. The EMBS generation system serves also as a visual interface prototyping system for non-computer specialist, which helps them overcome the initial learning barrier. The Program-Specification-by-Examples paradigm and visual programming by icons are key factors that facilitate the development of the above software. The users are allowed to specify the constraints and relationships among the cell values and the graphical elements displayed on the screen (e.g. a point, a line, line inclination), so that the cell values and graphical elements can be automatically updated as they are changed. As a case study of actual implementation, the paper describes the EMBS for determining air conditioning loads. In designing an air conditioning system, an equipment designer calculates heating and cooling loads and so forth for comfortable air conditioning. The EMBS can embody such users' expertise and offers and environment to perform their task more efficiently.

**Identifier:** **Smith90**  
**Title:** *ES-Kit: Rapid Prototyping of Scalable High Performance Systems*  
**Author:** Smith, K.S.  
**Source:** Advanced Research in VLSI. Proceedings of the Sixth MIT Conference; pp. 60-76; Cambridge, MA, USA; 2 April 1990 Cambridge, MA, USA; MIT Press; 1990; vi+398 pp. ISBN 0 262 04109 X  
**Date:** April, 1990  
**Keywords:** tools, real-time  
**Abstract:** An open kit of software and hardware building blocks is being developed at MCC, for application in rapid prototyping of innovative accelerators and other high performance parallel computing systems. Experimental Systems Kit (ES-Kit) hardware modules are small boards that can be stacked in very flexible three-dimensional arrangements, to implement scalable parallel architectures. Seven first generation module types became operational during 1989, including a 15 MIPS RISC processor, an 8 Mbyte memory, a 40 Mbyte/sec message communication co-processor, a message router motherboard, and a SCSI host adapter with integral 200 Mbyte magnetic disk. By late 1990, the ES-Kit project expects to complete development of at least a dozen different hardware module types. Architecturally insensitive supporting software modules now provided to users include a retargetable optimizing C++ compiler, parallel symbolic debugger, object-oriented OS kernel, librarian, dynamic linker, diagnostics, configurator, instrumentation data analysis tools, and an interactive front-end for Sun3 workstations. High performance experiments based on ES-Kit are already under development by collaborating research groups.

**Identifier:** Smyrniotis90a  
**Title:** *Rapid Prototyping: A Practitioner's Viewpoint in Software Development*  
**Author:** Smyrniotis, C.  
**Source:** SEKE '90. Proceedings. Software Engineering and Knowledge Engineering. 2nd International Conference; pp. 64-9; Skokie, IL, USA; 21-23 June 1990 Sponsored by: Knowledge Syst. Inst.; Inst. Inf. Ind.; S.W.I.F.T.; Univ. Pittsburgh Skokie, IL, USA; Knowledge Syst. Inst; 1990; iii+277 pp.  
**Date:** June, 1990  
**Keywords:** applications:miscellaneous, knowledge-based, process, methods, survey  
**Abstract:** In spite of all advances in conventional software development tools and methodologies, software development is still under the constant threat of the software crisis. The benefits of prototyping to software development have been known for a long time but only recent advents in artificial intelligence (AI) have made prototyping feasible for software development and have given rise to 'rapid prototyping'. The author has applied rapid prototyping to a number of diverse application areas including resource management, design validation, imagery exploitation, and requirements analysis. In the process he has found rapid prototyping to be a powerful approach for conceptualizing, defining, and building systems, while guarding against software crisis threats. The author examines rapid prototyping and its benefits to software development and offers guidelines for successful implementation. Six prototypes which have been built along with experience gained are discussed.

**Identifier:** Smyrniotis90b  
**Title:** *Rapid Prototyping: A Cure for Software Crisis*  
**Author:** Smyrniotis, C.  
**Source:** Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences; pp. 202-10 vol.2; Kailua-Kona, HI, USA; 2-5 Jan. 1990 Sponsored by: IEEE; Univ. Hawaii; PRIISM; ACM Los Alamitos, CA, USA; IEEE Comput. Soc. Press; 1990; 4 vol. (x+449+xii+575+xii+673+xi+515) pp.  
**Date:** January, 1990  
**Keywords:** process, knowledge-based, applications:miscellaneous  
**Abstract:** The failure to produce reliable software within a reasonable time and cost and the predicted shortage of software engineers to meet demands for new software have given rise to the so-called software crisis. Rapid prototyping was applied to a number of diverse application areas as the applicability of AI (artificial intelligence) technology was explored. Prototypes have been developed in resource management and scheduling, fluid network design and validation, and imagery exploitation. The authors examine the benefits of rapid prototyping in software development and discuss prototypes they have built and experience they have gained.

**Identifier:** Son88  
**Title:** *A Prototyping Environment for Distributed Database Systems*  
**Author:** Son, S.H.; Kim, Y.  
**Source:** Computer science report / University of Virginia. School of Engineering and Applied Science. Dept. of Computer Science; no. TR-88-20 Computer science report (University of Virginia. Dept. of Computer Science); no. TR-88-20.  
**Date:** 1988  
**Keywords:** tools, distributed processing, database  
**Abstract:** This paper describes a software prototyping environment for the development and evaluation of distributed database systems. The prototyping environment is based on concurrent programming kernel which supports the creation, blocking, and termination of processes, as well as scheduling and interprocess communication. The paper proposes the port construct to represent a flexible and modular message-communication facility. A general blocking construct is used for process scheduling and message-communication in simulated time. Based on these two notions, the paper describes the prototyping environment that has been developed and a series of experimentation performed for performance evaluation of a multiversion database system. One of the key aspects of the prototyping environment, can be easily ported to a target hardware system for embedded testing.

**Identifier:** Tamanaha90  
**Title:** *Rapid Prototyping of Large Command, Control, Communications and Intelligence (C<sup>3</sup>I) Systems*  
**Author:** Tamanaha, D.Y.; Bourgeois, P.J.  
**Source:** 1990 IEEE Aerospace Applications Conference Digest (Cat. No.90TH0223-8); pp. 253-63; Vail, CO, USA; 4-9 Feb. 1990 Sponsored by: IEEE New York, NY, USA; IEEE; 1990; iv+318 pp.  
**Date:** February, 1990  
**Keywords:** real-time, process, methods, structured analysis, applications: C<sup>3</sup>I  
**Abstract:** Rapid prototyping is examined from three points of view: management, rapid analysis, and design. A rapid prototyping approach is presented for end-user requirements of large, data-intensive, command and control (C<sup>2</sup>); command, control, communications and intelligence (C<sup>3</sup>I); and command and control information systems (CCIS). It is noted that participatory management, highly motivated personnel, thorough knowledge of the targeted system's operations, sound prototyping methodology, appropriate tools, and innovative techniques used by an integrated team allow for severe schedule constraints and provide the edge for fast implementation. Modified structured methods and further innovations allow a rapid prototype cycle. Experience gained and examples are cited to illustrate the ideas and methods used in successful C<sup>3</sup>I and CCIS prototypes. The authors discuss the compressed usage of known methods and introduce an innovative design method for the rapid development of operational threads as an integrating design technique to quickly assemble knowledge of disparate design views and disciplines under severe prototyping schedule constraints

**Identifier:** Tenazas90  
**Title:** *DPSOI: An Executable Requirements Specification Language for Information Processing Systems*  
**Author:** Tenazas, R.A.; Concepcion, A.I.; Villafuerte, R.M.  
**Source:** Proceedings of the Twenty-Third Annual Hawaii International Conference on System Sciences; pp. 47-54 vol.2; Kailua-Kona, HI, USA; 2-5 Jan. 1990  
Sponsored by: IEEE; Univ. Hawaii; PRIISM; ACM Los Alamitos, CA, USA; IEEE Comput. Soc. Press; 1990; 4 vol. (x+449+xii+575+xii+673+xi+515) pp.  
**Date:** January, 1990  
**Keywords:** languages, tools, executable specifications  
**Abstract:** The authors present the Delphi project, which deals with the design and implementation of a software engineering environment (SEE) to enhance software productivity, in particular, one that targets information systems (IS) for business data processing problems. A set of specification languages was designed to specify the IS in a modular manner. Collectively, these languages are called the data processing specification outlining language (DPSOL). DPSOL is customizable, executable, extensible, and translatable and can be used in rapid prototyping or in automatic translation to a target 3GL/environment. The authors present the expressive power of DPSOL to specify the IS, and its automated process of checking referential integrity and computational consistency. The approach is translational (using templates) instead of transformational. Templating is a powerful concept in automatic programming, at least for IS. The templates can be edited and modified to generate customized 3GL source code. Moreover, documentation can also be automatically generated

**Identifier:** Thayer90  
**Title:** *System and Software Requirements Engineering*  
**Author:** Thayer, R.H.; Dorfman, M.  
**Source:** Book  
**Date:** 1990  
**Keywords:** collection  
**Notes:** This IEEE tutorial includes seminal works on the following topic areas: requirements engineering for systems and software, system and software engineering, software requirements analysis and specification, software requirements methodologies and representation methods, software requirements engineering tools and techniques, requirements and quality management, software systems engineering process models, and case studies. Also included is an extensive glossary and annotated bibliography.

**Identifier:** Trenouth91  
**Title:** *A Survey of Exploratory Software Development*  
**Author:** Trenouth, J.  
**Source:** Computer Journal; vol.34, no.2; April 1991; pp. 153-63  
**Date:** April, 1991  
**Keywords:** process, methods, survey  
**Abstract:** Exploratory software development is an important style of software development that has a markedly different flavor from conventional software engineering methodologies. Originally used in artificial intelligence programming, it has much in common with both rapid prototyping and software maintenance. The paper surveys the area, by examining the methodology, technology, and related issues.

**Identifier:** Tsai89  
**Title:** *Knowledge-based System for Rapid Prototyping*  
**Author:** Tsai, J.J.P.; Lie, A.  
**Source:** Knowledge-Based Systems; vol.2, no.4; Dec. 1989; pp. 239-48  
**Date:** December, 1989  
**Keywords:** knowledge-based, process, languages  
**Abstract:** Rapid prototyping produces better software products and research on combining artificial intelligence and software engineering has been conducted for a number of years. A knowledge-based system for rapid prototyping is presented. In the system, the Frame-and-Rule Oriented Requirements Language and a methodology are developed to provide an integrated means of prototyping throughout the software life cycle. The particular application domain to be modelled is represented in terms of objects and activities. FRORL, which uses the concept of frames and production systems, describes the problem domain's objects and activities in a natural way. With the support of a knowledge base, a software prototype can be rapidly developed using FRORL. The system has been implemented using Prolog on a VAX-11/780 computer.

**Identifier:** Tsai90a  
**Title:** *A Knowledge-based Approach to Rapid Prototyping Systems*  
**Author:** Tsai, S.T.; Yang, C.C.; Lien, C.C.  
**Source:** Journal of the Chinese Institute of Engineers; vol.13, no.5; Sept. 1990; pp. 505-18  
**Date:** September, 1990  
**Keywords:** knowledge-based, tools, languages  
**Abstract:** A developing system for supporting the construction of software prototypes should provide some automatic tools in order that the prototypes can be generated in shorter time and with lower cost. The authors present an integrated knowledge-based rapid prototyping system (KBRPS). This system contains a graphic conceptual model for describing system behaviors; a frame-based software requirements specification language (FSRSL) to represent the internal forms of the conceptual model and to further specify detailed activities and constraints; a database for storing specification files; and a knowledge base for storing rules of specification analysis and specification transformation. The specification analysis can check the consistency and completeness of requirements specifications. Specifications written in FSRSL can also be executed as a software prototype. If the prototype specifications meet the user's requirements, they can be automatically transformed into C language programs; otherwise the original specifications can be modified iteratively until a satisfactory prototype or system is obtained.

**Identifier:** Tsai90b  
**Title:** *Automated Retrieval of Consistent Documentation for Rapid Prototyping Systems and Software Maintenance*  
**Author:** Tsai, S.T.; Yang, C.C.; Lien, C.C.  
**Source:** Information and Software Technology; vol.32, no.8; Oct. 1990; pp. 521-30  
**Date:** October, 1990  
**Keywords:** documentation, knowledge-based, languages  
**Abstract:** As effective software understanding is dependent on the correctness and clearness of system documentation, automated support for the documents of software prototypes and maintenance clearly points to the use of database technology. A knowledge-based rapid prototyping system (KBRPS) has been developed that effectively helps the construction of software prototypes. KBRPS contains a graphic representation of the conceptual model for modelling system structures, a frame-based software requirements specification language (FSRSL) for describing the textual form of the conceptual model and specifying detailed system behaviors, a database for stored specifications files, and a knowledge base for stored rules of software development. The system emphasizes that, first, software development or maintenance should comprehend the real system requirements, then the requirements specifications can be transformed into program code. The FSRSL specifications stored in the database can be retrieved by a query system for generating formal documents. These documents, which are helpful in understanding the developed system for prototype modification and software maintenance, include the conceptual model, FSRSL specifications, the abstract relations of hierarchical specifications, and even answers to particular questions.

**Identifier:** Tucheran90  
**Title:** *The CHRIS Consultant- A Tool for Database Design and Rapid Prototyping*  
**Author:** Tucheran, L.; Casanova, M.A.; Furtado, A.L.  
**Source:** Information Systems; vol.15, no.2; 1990; pp. 187-95  
**Date:** 1990  
**Keywords:** tools, database, knowledge-based  
**Abstract:** CHRIS is an expert software tool to help in the design and rapid prototyping of information systems containing a database component. CHRIS involves an extended entity-relationship information model, the relational data model and a database management system. A prototype version of the tool, written in Prolog extended with a query-the-user facility, is fully operational. The prototype includes an interface for experimental use which enforces the integrity constraints of the application.

**Identifier:** Turnhelm89  
**Title:** *Rapid Prototyping of the Operational Definition of Command and Control Consoles*  
**Author:** Turnheim, A.; Lachaover, I.  
**Source:** Proceedings. Fourth Israel Conference on Computer Systems and Software Engineering (Cat. No.89CH2660-9); pp. 133-8; Herzlia, Israel; 5-6 June 1989  
Sponsored by: IEEE Washington, DC, USA; IEEE Comput. Soc. Press; 1989;  
iii+192 pp. ISBN 0 8186 1972 4  
**Date:** June, 1989  
**Keywords:** tools, knowledge-based, user-interface  
**Abstract:** A description is given of a software package that defines the operational modes of a command and control console using an interactively editable simulation of this operation. The package runs on a TI-Explorer Lisp machine and uses rapid-prototyping techniques and AI tools and methodology. The future prospects of similar programs as lifetime support for small command and control systems are predicted, showing possible support in the areas of sales, definition, coding, integration, and maintenance.

**Identifier:** Tyszberowicz89  
**Title:** *OBSERV: A Prototyping Language and Environment Combining Object Oriented Approach, State Machines and Logic Programming*  
**Author:** Tyszberowicz, S.; Yehudai, A  
**Source:** Computer science technical report series / University of Maryland; CS-TR-2304  
Computer science technical report series (University of Maryland at College Park); CS-TR-2304.  
**Date:** August, 1989  
**Keywords:** languages, methods, formalism

**Identifier:** Wallentinson89  
**Title:** *Rapid Prototyping in Command and Control System Development*  
**Author:** Wallentinson, C.  
**Source:** Conference Proceedings MILCOMP 89, Military Computers Systems and Software; pp. 70-1; London, UK; 26-28 Sept. 1989 Tunbridge Wells, UK; Microwave Exhibitions & Publishers; 1989; 425+22 pp. ISBN 0 946821 86 0  
**Date:** September, 1989  
**Keywords:** user-interface, tools  
**Abstract:** Rapid prototyping is a method that starts with the man-machine interface and builds a prototype system that from the operator's point of view acts like a real system. This paper describes a rapid prototyping system based on commercial PC computers and high resolution color monitors. The system is designed to fulfil the demands of small to medium scale command and control systems

**Identifier:** Warkowski90  
**Title:** *IC<sup>3</sup>: A Neural ASIC for Real-time Prototyping*  
**Author:** Warkowski, F.; Spaanenburg, L.; Nijhuis, J.A.G.  
**Source:** Parallel Processing in Neural Systems and Computers; pp. 319-22; Dusseldorf, West Germany; 19-21 March 1990 Sponsored by: Robert Bosch; IBM; Philips; Siemens; et al Amsterdam, Netherlands; North-Holland; 1990; xv+626 pp. ISBN 0 444 88390 8  
**Date:** March, 1990  
**Keywords:** applications: neural network, simulation  
**Abstract:** The architecture and application of a master-slice concept for rapid prototyping of neural networks is presented. After simulation with the NNSIM neural network simulation environment a netlist specifies the interconnection on a master of neural building blocks, which can be personalized to various architectures with metallization; while maintaining a range of options for mask-and software programmability.

**Identifier:** Wellner89  
**Title:** *Statemaster: A UIMS Based on Statecharts for Prototyping and Target Implementation*  
**Author:** Wellner, P.D.  
**Source:** SIGCHI Bulletin; spec. issue.; May 1989; pp. 177-82 Conference on Human Factors in Computing Systems (CHI 89); Austin, TX, USA; 30 April-4 May 1989 Sponsored by: IEEE; ACM  
**Date:** May, 1989  
**Keywords:** user-interface, tools  
**Abstract:** Most user interface management systems are state based and some use state transition diagrams for dialog specification. Although these diagrams have significant advantages, they suffer from drawbacks that make them impractical for the specification of complex user interfaces. Statecharts are a hierarchical extension of state transition diagrams and are well suited for specification of complex user interface dialogs. Statemaster is a UIMS implemented in C++ that uses statecharts for dialog specification. It has been successfully used both for rapid prototyping and target implementation of user interfaces. The paper describes the use of statecharts for dialog specification and the implementation of Statemaster.

**Identifier:** Whatmore91  
**Title:** *Simulation of Modern Electronic Combat Scenarios by Means of a Flexible Generic Computer Model*  
**Author:** Whatmore, L.C.; Smith, A.M.  
**Source:** IEE Colloquium on 'Electronic Warfare Systems' (Digest No.009); pp. 10/1-5; London, UK; 14 Jan. 1991 Sponsored by: IEE London, UK; IEE; 1991; 54 pp.  
**Date:** January, 1991  
**Keywords:** simulation, real-time, parameterized models, applications: electronic warfare  
**Abstract:** The effectiveness of EW equipment and tactics is becoming more difficult to predict as the level of complexity of ECCM and ECM increases. Simulation, when validated, is an important tool in the evaluation of EW equipment and tactics. It complements traditional methods such as laboratory measurements or trials, providing a cost effective, secure and repeatable means of extrapolating into dense scenarios, performing sensitivity analysis, rapid prototyping of proposed enhancements, quantitative comparisons of alternative designs, and performance evaluation against current or postulated threats. To be most cost-effective, a tool should be flexible. A generic simulation model, with parametric functional models of equipment, provides maximum flexibility if it is based on a framework which imposes no unnecessary limitations. The requirements for such a framework have been given and an implementation of a pulse-by-pulse functional model by Software Sciences outlined. Sample applications of this model and others in the family have been given to demonstrate the simulation of modern electronic combat scenarios by means of a flexible generic computer model.

**Identifier:** Wing91  
**Title:** *Unintrusive Ways to Integrate Formal Specifications in Practice*  
**Author:** Wing, J.M.; Moormann Zaremski, A.  
**Source:** Research paper. Carnegie Mellon University, Computer Science Dept.; CMU-CS-91-113  
**Date:** 1991  
**Keywords:** formalism, methods, structured analysis  
**Abstract:** Formal methods can be neatly woven in with less formal, but more widely-used, industrial-strength methods. We show how to integrate the Larch two-tiered specification method [GHW85a] with two used in the waterfall model of software development: Structured Analysis [Ros77] and Structure Charts [YC79]. We use Larch traits to define data elements in a data dictionary and the functionality of basic activities in Structured Analysis data-flow diagrams; Larch interfaces and traits to define the behavior of modules in Structure Charts. We also show how to integrate loosely formal specification in a prototyping model by discussing ways of refining Larch specifications as code evolves. To provide some realism to our ideas, we draw our examples from a non-trivial Larch specification of the graphical editor for the Miro visual languages [HMT+90]. The companion technical report, CMU-CS-91-111, contains the entire specification.

**Identifier:** Zhao91  
**Title:** *An Environment for Rapid Prototyping of Interactive Systems*  
**Author:** Zhao, J.; Liu, S.  
**Source:** Journal of Computer Science and Technology (English Language Edition); vol.6, no.2; April 1991; pp. 135-44  
**Date:** April, 1991  
**Keywords:** user-interface, tools, methods, languages  
**Abstract:** The paper shows an environment which supports the development of multi-thread dialogue interactive systems. The environment includes several tools and run-time support programs for the design and implementation of the user interface of an interactive system. First, methods of user interface specification with elementary nets are discussed. Then, the syntax of a user interface specification language based on elementary nets and the pre-compiler for the language as well as a graphic editor for elementary nets construction are described. Finally, an example is given to illustrate the design process of a user interface.

**Identifier:** Zompi90  
**Title:** *Rapid Prototyping through Graphical Operational Specification and Automated Code Generation*  
**Author:** Zompi, R.; Russi, V.  
**Source:** COMPEURO '90. Proceedings of the 1990 IEEE International Conference on Computer Systems and Software Engineering (Cat. No.90CH2867-0); pp. 509-17; Tel-Aviv, Israel; 8-10 May 1990 Sponsored by: IEEE; Inf. Processing Assoc. Israel Los Alamitos, CA, USA; IEEE Comput. Soc. Press; 1990; xiii+574 pp. ISBN 0 8186 2041 2  
**Date:** May, 1990  
**Keywords:** graphical specification, tools, object-oriented, distributed processing  
**Abstract:** The application of PROTOB, an object-oriented CASE system based on high-level Petri nets, to rapid prototyping of distributed systems is presented. PROTOB consists of several tools supporting specification, modeling and prototyping activities within the framework of the operational software life-cycle paradigm. As its major application area it addresses distributed systems, such as real-time embedded systems, communication protocols and manufacturing control systems. The PROTOB methodology and its support environment can be used in software development of distributed discrete-event dynamic systems at three different levels. The functionality of the system can be formally defined and also analyzed quantitatively by building a PROT net based model, which is actually a simulation model. The model becomes more detailed and the timing of the transitions is real, being managed by the host operating system. The PROTOB objects emulating the physical environment are replaced by a suitable interface which has the task of transforming signals coming from the plant into tokens to be introduced into the PROTOB model and, likewise, of converting tokens coming from the PROTOB model into appropriate commands issued to the plant.

## Index by Author

Alpuente, M. 37  
Andrulis, M. W. 24  
Aoyama, M. 30

Bagrodia, R.L. 25  
Bailes, P. A. 25  
Bajwa, L. Y. 25  
Barbacci, M. R. 26  
Barnes, P.D. 48  
Bartschi, M. 27  
Baumann, R. 36  
Beradi, L. 27  
Berzins, V. 48  
Biggie, A.V.L. 28  
Birch, M. 28  
Black, H. 29  
Bourgeois, P.J. 60  
Brooks, F.P. Jr. 29  
Buchanan, W.E. 28

Card, R. 44  
Casanova, M.A. 64  
Cassamayor, J.C. 37  
Chang, C.K. 30  
Chau, P. 41  
Chung, C.G. 46  
Clark, J. 54  
Cohen, D.M. 30  
Concepcion, A.I. 61  
Cooling, J.E. 31, 42  
Cooper, S. 31  
Cordy, J.R. 32  
Crespo, A. 35

Dalesio, L.R. 45  
Daley, P.C. 57  
de la Puente, J.A. 35  
Degl'Innocenti, M. 33  
Demeure, I.M. 33  
DeSoi, J.F. 32  
Diaz-Gonzalez, J.P. 34  
Dorfman, M. 61  
Downs, T. 35  
DuPont, F.G. 47

Ebert, J. 36  
Edmonson, D. 34  
Ekambareshwar, S. 35  
Elzer, P. 51  
Espinosa, A. 35  
Estraillier, P. 44

Ferrari, G.L. 33  
Fisher, G.E. 35  
Forger, A.F. 38  
Furtado, A.L. 64

Ganti, M. 36  
Garcia-Fornes, A. 35  
Gerber, S. 36  
Gimnich, R. 36  
Giordano, F. 37  
Gomaa, H. 37  
Gonzalez de Rio, A. 37  
Goyal, P. 36  
Gregorio, D.D. 38  
Guinther, T.M. 30  
Gutierrez, O. 38

Halpern-Hamu, C.D. 32  
Harris, J.R. 39  
Hartson, H.R. 39  
Hawryszkiewicz, I.T. 40  
Hazan, P.L. 28  
Heisler, K.G. 41  
Hekmatpour, A. 41  
Henskes, D.T. 41, 42  
Hill, J.O. 45  
Hughes, G. 48  
Hughes, T.S. 31, 42

Jain, A.K. 43  
Johnson, P. 34  
Jones, M.L.R. 43

Kossiakoff, A. 28  
Kozubal, A.J. 45  
Kreutzer, W. 45, 46  
Krista, R. 46

Halpern-Hamu, C.D. 32  
Harris, J.R. 39  
Hartson, H.R. 39  
Hawryszkiewicz, I.T. 40  
Hazan, P.L. 28  
Heisler, K.G. 41  
Hekmatpour, A. 41  
Henskes, D.T. 41, 42  
Hill, J.O. 45  
Hughes, G. 48  
Hughes, T.S. 31, 42

Jain, A.K. 43  
Johnson, P. 34  
Jones, M.L.R. 43

Kossiakoff, A. 28  
Kozubal, A.J. 45  
Kreutzer, W. 45, 46  
Krista, R. 46  
Kunii, T.L. 58

Lachaover, I. 64  
Larson, A. 54  
Lea, R.J. 46  
Leciston, D. 29  
Lichota, R. W. 26  
Lie, A. 62  
Lien, C.C. 63  
Liu, S. 67  
Lively, W.M. 32  
Luckey, P.H. 47  
Luqi 47, 48

Madison, D.E. 49  
Matsumoto, Y 49  
McArdle, B.R. 38  
McCollum, L. 37  
McEnery, A.M. 50  
McGhee, R. 29  
McInroy, J.W. 50  
Milovanovic, R. 51  
Minkowitz, C. 51  
Moody, S.A. 52  
Moormann Zaremski, A. 67

Nassif, R. 36  
Naveda, F. 48  
Ness, L.A. 30  
Nijhuis, J.A.G. 65  
Nugent, E.R. 52  
Nutt, G.J. 33

O'Neil, G. 52  
Oakes, M.P. 50  
Ohno, Y. 49  
Ortner, E. 53  
Overmyer, S.P. 53

Pacini, G. 33  
Parker, D.W. 39  
Pastor, M.A. 37  
Podar, S. 36  
Powers, M.K. 54  
Promislow, E. 32  
Purtilo, J. 54

Ramirez, M.J. 37  
Ramos, I. 37  
Reid, D. 50  
Rieche, B. 27  
Rizman, K. 55  
Rohrle, J. 53  
Royce, W. 56  
Rozman, I. 46, 55  
Russi, V. 68  
Rzepka, W.E. 57

Salzman, E. J. 25  
Scott, D.B.H. 37  
Sheepard, S.V. 32  
Shirota, Y. 58  
Smith, A.M. 66  
Smith, E.C. 39  
Smith, K.S. 58  
Smyrniotis, C. 59  
Son, S.H. 60  
Spaanenburg, L. 65  
Steigerwald, R. 48

Tamanaha, D.Y. 60  
Tenazas, R.A. 61  
Thayer, R.H. 61  
Tink, P.D. 43  
Tolmie, J.C. 41, 42  
Trenouth, J. 62  
Tresch, M. 27  
Tsai, J.J.P. 62  
Tsai, S.T. 63  
Tsai, W.T. 41  
Tucherman, L. 64  
Turini, F. 33  
Turnheim, A. 64  
Tyszberowicz, S. 65

Urban, J.E. 34

Verber, D. 55  
Villafuerte, R.M. 61

Wallentinson, C. 65  
Warkowski, F. 65  
Welch, T.A. 44  
Wellner, P.D. 66  
Whatmore, L.C. 66  
Whiteley, K. 28  
Wing, J.M. 67  
Wong, B. 37

Yang, C.C. 63  
Yehudai, A 65  
Young-Fu Chang 30

Zhao, J 67  
Zimmerlich, J. 29  
Zompi, R. 68  
Zyda, M. 48



## Index by Keyword

Ada 24, 25, 35, 47, 56

applications

  avionics 27

  C2 37

  C3I 38, 52, 60

  communications 30

  control systems 45

  education 28

  electronic warfare 66

  environment 25

  health care 39, 49

  industrial 31

  miscellaneous 59

  networks 43, 56

  neural network 65

  process control 36, 37

  programming languages 32

  telephony 30, 44

collection 50, 57, 61

conceptual modeling 38, 40

database 27, 40, 47, 50, 53, 60, 64

distributed processing 25, 26, 30, 33, 45,  
56, 60, 68

documentation 63

executable specifications 26, 30, 33, 38,  
44, 46, 49, 51, 53, 61

formalism 25, 33, 34, 35, 37, 44, 52, 53, 65,  
67

graphical specification 27, 32, 33, 41, 45,  
46, 48, 49, 55, 68

knowledge-based 27, 28, 32, 36, 41, 45, 49,  
51, 55, 59, 62, 63, 64

languages 30, 34, 35, 44, 55, 61, 62, 63, 65,  
67

logic programming 33, 49

methods 24, 25, 26, 27, 29, 32, 34, 39, 40,  
42, 43, 44, 46, 48, 52, 53, 55, 56, 57,

58, 59, 60, 62, 65, 67

object-oriented 24, 27, 31, 34, 36, 41, 45, 52, 68

parameterized models 66

process 24, 25, 29, 37, 43, 48, 49, 54, 59, 60,  
62

real-time 25, 26, 28, 29, 30, 31, 33, 34, 35, 37,  
38, 42, 44, 48, 49, 57, 58, 60, 66

reuse 30, 36, 44, 47, 56, 57

simulation 30, 33, 34, 38, 41, 42, 45, 46, 52, 65,  
66

structured analysis 26, 42, 46, 55, 60, 67

survey 24, 29, 31, 36, 39, 43, 50, 51, 59, 62

tools 27, 28, 29, 31, 32, 33, 34, 35, 36, 38, 41,  
42, 43, 44, 45, 46, 48, 49, 50, 51, 52, 54,  
55, 56, 57, 58, 60, 61, 63, 64, 65, 66, 67,  
68

user-interface 29, 31, 32, 38, 39, 40, 41, 42, 49,  
52, 54, 57, 58, 64, 65, 66, 67



## Index by Source

### Book

AGARD Lecture Series 27  
Japanese Perspectives in Software Engineering 50  
Sample CASE Tools and Perspectives in CASE 44  
Simulation and the User Interface 34  
Software Engineering for Large Software Systems 52  
System and Software Requirements Engineering 61

### Periodical

Case Outlook 90 29  
Computer Journal 62  
Computer Languages 32, 34  
Computer Science And Technology 36  
Computers in Healthcare 49  
Der Elektroniker 36  
Electrical Communication 42  
Elektrotehniski Vestnik 56  
IEE Colloquium on 'Electronic Warfare Systems' 66  
IEE Colloquium on 'Software Tools for Interface Design' 43  
IEE Colloquium on 'Specification of Complex Systems' 42  
IEEE Computer 29, 48  
IEEE Transactions on Software Engineering 33  
Information and Software Technology 46, 49, 63  
Information Systems 64  
Interacting with Computers 40  
Johns Hopkins APL Technical Digest 28  
Journal of Computer Science and Technology 67  
Journal of Object-Oriented Programming 46  
Journal of the Chinese Institute of Engineers 63  
Knowledge-Based Systems 62  
SIGCHI Bulletin 32, 39, 66  
Signal 24, 37  
SIGSOFT Software Engineering Notes 46, 54

### Simulation 30

Systems International 31

### Proceedings

12th BCS IRSG Research Colloquium on Information Retrieval 50  
1988 Summer Computer Simulation Conference 52  
1990 IEEE Aerospace Applications Conference Digest 60  
1990 IEEE International Conference on Computer Systems and Software Engineering 35, 68  
1990 Summer Computer Simulation Conference 38, 45  
2nd IFAC/IFIP Workshop 43, 51, 53  
2nd International Conference 59  
ACM SIGGRAPH Symposium on User Interface Software and Technology 54  
Advanced Research in VLSI 58  
Australian Computer Conference - 1987 40  
COMPCON Spring '90 36  
COMPEURO '90 35, 68  
Conference on Computing Systems and Information Technology 1989 35  
Experience with the Management of Software Projects 1988 43, 53  
First International Conference on Systems Integration 30  
Fourth Israel Conference on Computer Systems and Software Engineering 64  
GLOBECOM '89 43  
IATED International Symposium 38  
IEE Colloquium on 'Specification of Complex Systems' 42  
IEEE 13th International Conference on Software Engineering 55  
IEEE 1989 NAECON 25  
IEEE Global Telecommunications Conference and Exhibition. Communications Technology for the 1990s and Beyond 43  
INTERACT '87 37, 39, 41  
International Conference on Accelerator and Large Experimental Physics Control Systems 45

MILCOMP 89 65  
Military Computers Systems and  
Software 65  
Nineteenth Annual Hawaii International  
Conference on Systems  
Sciences 57  
Ninth Annual International Phoenix  
Conference on Computers and  
Communications 44  
Parallel Processing in Neural Systems  
and Computers 65  
Second IFIP Conference 37, 39, 41  
Second International Conference on  
Software Engineering for Real  
Time Systems 28, 31  
SEI Requirements Engineering and  
Analysis Workshop 49  
SEKE '90 55, 57, 59  
Seventh Washington Ada Symposium  
47  
Sixth MIT Conference 58  
Software Engineering and Knowledge  
Engineering 59  
Software Engineering and Knowledge  
Engineering. 2nd International  
Conference 55, 57  
SPIE - The International Society for  
Optical Engineering 41, 51  
Systems Integration '90 25, 30  
Technology of Object-Oriented  
Languages and Systems 27  
Thirty-Fifth IEEE Computer Society  
International Conference 36  
TOOLS '89 27  
TRI-Ada '89 56  
Twenty-Third Annual Hawaii  
International Conference on  
System Sciences 48, 59, 61

Technology 41  
National Physics Laboratory 53  
Naval Postgraduate School 48  
Queensland, University of, Department of  
Computer Science 25  
Tokyo, University of, Dept. of Information  
Science 58  
University of Minnesota (dissertation) 48  
Virginia, University of, Dept. of Computer  
Science 60

#### Technical Report

Carnegie Mellon University, Computer  
Science Dept. 67  
CECOM-TR-90-2 29  
Colorado, Boulder, University of, Dept.  
of Computer Science 33  
IEEE CH1627-9/81/0000/0333 37  
Maryland, University of 65  
Minnesota, University of, Institute of

## Index by Title

- Acquisition Model for the Capture and Management of Requirements for Battlefield Software Systems 29
- AI Techniques and Object-oriented Technology for VLSI Design-space Representation, Optimization and Management 41
- Application Software Prototyping and Fourth Generation Languages 36
- Automated Retrieval of Consistent Documentation for Rapid Prototyping Systems and Software Maintenance 63
- Between Man and Machine 31
- CAPS as a Requirements Engineering Tool 49
- CHRIS Consultant- A Tool for Database Design and Rapid Prototyping, The 64
- Computer Aided Prototyping Methodology, A 46
- Constructive Formal Specifications for Rapid Prototyping 37
- DETAIL: An Approach to Task Analysis 34
- Dictionary-supported Prototyping of Database Applications 53
- Display Rapid Prototyping and Simulation System, A 38
- Distributed Interactive Scenario Generator for Command, Control And Communications, A 52
- DPSOI: An Executable Requirements Specification Language for Information Processing Systems 61
- Durra: An Integrated Approach to Software Specification, Modeling and Rapid Prototyping 26
- Emergence of Rapid Prototyping as a Real-time Software Development Tool, The 31, 42
- Ensemble: A Graphical User Interface Development System for the Design and Use of Interactive Toolkits 54
- Environment for Rapid Prototyping of Interactive Systems, An 67
- ES-Kit: Rapid Prototyping of Scalable High Performance Systems 58
- Evaluation of Rapid Prototyping Methodology in a Human Interface 39
- Experience with the Management of Software Projects 1988 51
- Functional Prototyping with Proto 44
- Graphical Specification of User Interfaces with Behavior Abstraction 32
- Graphical Support for Reducing Information Overload in Rapid Prototyping 48
- Graphical Tool for Computer-aided Prototyping 49
- HyperCard- The Legend: Summary 43
- IC3: A Neural ASIC for Real-time Prototyping 65
- IEE Colloquium on 'Specification of Complex Systems' (Digest No.145) 42
- Impact of DoD-Std-2167A on Iterative Design Methodologies: Help or Hinder?, The 54
- Information Systems Design and Prototyping Using an Object-oriented

Software Engineering Environment 27  
Integrated Approach to the Design and Performance Evaluation of Distributed Systems, An 25  
Integrating the Role of Rapid Prototyping and Requirements Specification Using the  
Object-oriented Paradigm 41

Japanese Perspectives in Software Engineering 50

Knowledge Base Approach to the Specification Of Real Time System Requirements, A 28  
Knowledge-based Approach to Rapid Prototyping Systems, A 63  
Knowledge-based Software in a Realtime Alarm-handling System 36  
Knowledge-based System for Rapid Prototyping 62

Language Aspects of ENVISAGER: an Object-oriented Environment for the  
Specification of Real-time Systems 34  
Lessons Learned from the Use of a Spiral Model for an Ada Development Effort  
the Software Life Cycle Support Environment (SLCSE) 25

Methodical and Management Experiences from an Extensive Software Project 43  
Methodology for Prototyping-in-the-Large, A 55  
Modeller's Assistant- A First Step Towards Integration of Knowledge Bases and  
Modelling Systems, The 45  
Multimedia Rapid Prototyping Tool for the Development Of Computer-assisted Instruction, A  
28

No Silver Bullet: Essence and Accidents of Software Engineering 29

Object-oriented Application Development Environment, An 36  
Object-oriented Development Aids Prototyping and Delivery 24  
OBSERV: A Prototyping Language and Environment Combining Object Oriented Approach,  
State Machines and Logic Programming 65

Pantheon: Rapid Prototyping of Natural Language Interfaces to Large Databases 50  
Protolog: A Conceptual Schema Facility for Automated Prototype Generation 38  
Prototyping and Simulating Parallel, Distributed Computations with VISA 33  
Prototyping and Visualisation in Interface Design 42  
Prototyping as a Tool in the Specification of User Requirements 37  
Prototyping Environment for Distributed Database Systems, A 60  
Prototyping Techniques for Different Problem Contexts 39  
Prototyping Tool to Assist in Requirements Engineering, A 57  
Prototyping with the Entity-relationship Model 40

QUISAP: An Environment for Rapid Prototyping of Real-time Systems 35

Rapid Ada Prototyping: Principles and Example of a Complex Application 44  
Rapid Development Speeds Path for Command System 37

- Rapid Prototyping Approach to Software Development and our Tool for Development of RPT Prototypes, A 56
- Rapid prototyping Capabilities in the Expert Requirements Expression and System Synthesis (EXPRESS) Environment 51
- Rapid Prototyping for Healthcare Applications 49
- Rapid Prototyping for Large Software System Design 48
- Rapid Prototyping from Structured Analysis: Executable Specification Approach 46
- Rapid Prototyping in Ada in the Rational Environment Emphasizing Software Reuse 47
- Rapid Prototyping in Command and Control System Development 65
- Rapid Prototyping in Human-Computer Interface Development 40
- Rapid Prototyping of Communications Protocols Using a New Parallel Language 30
- Rapid Prototyping of Complex Avionics Systems 27
- Rapid Prototyping of Formal Specifications Using Miranda 53
- Rapid Prototyping of Large Command, Control, Communications and Intelligence (C3I) Systems 60
- Rapid Prototyping of Man-machine Interfaces for Telecommunications Equipment Using Interactive Animated Computer Graphics 41
- Rapid Prototyping of Software Systems Using Prolog 35
- Rapid Prototyping of the Operational Definition of Command and Control Consoles 64
- Rapid Prototyping through Graphical Operational Specification and Automated Code Generation 68
- Rapid Prototyping: A Cure for Software Crisis 59
- Rapid Prototyping: A Practitioner's Viewpoint in Software Developmen 59
- Real-time Distributed Simulation of Pbx with Software Reuse, A 30
- Reliable, Reusable Ada Components for Constructing Large, Distributed Multi-task Networks: Network Architecture Services (NAS) 56
- Research Aspects of Rapid Prototyping 48
- RSF: a Formalism for Executable Requirement Specifications 33
- Run-time Environment and Application Tools for the Ground Test Accelerator Control System 45
- SEKE '90. Proceedings. Software Engineering and Knowledge Engineering. 2nd International Conference 57
- Simulation of Modern Electronic Combat Scenarios by Means of a Flexible Generic Computer Model 66
- Software Architecture Modelling 52
- Software Development by Functional Language Prototyping 25
- Software Evolution through Rapid Prototyping 48
- Software Quality via Rapid Prototyping 43
- Specification and Automatic Generation of Intelligent Graphical Interfaces 58
- Statemaster: A UIMS Based on Statecharts for Prototyping and Target Implementation 66
- Survey of Exploratory Software Development, A 62
- Survey of Rapid Prototyping Tools 29
- System and Software Requirements Engineering 61
- Tiny Tim-a Smalltalk Toolbox for Rapid Prototyping and Animation of Models 46
- TXL: A Rapid Prototyping System for Programming Language Dialects 32

Unintrusive Ways to Integrate Formal Specifications in Practice 67  
Using Data-flow Description Supported by the Rapid Prototyping Tool for  
Specifying and Developing of Knowledge-based Systems 55

## Index by Year

1981

Prototyping as a Tool in the Specification of User Requirements 37

1986

Prototyping Tool to Assist in Requirements Engineering, A 57

Rapid Prototyping for Large Software System Design 48

1987

Application Software Prototyping and Fourth Generation Languages 36

Constructive Formal Specifications for Rapid Prototyping 37

Evaluation of Rapid Prototyping Methodology in a Human Interface 39

No Silver Bullet: Essence and Accidents of Software Engineering 29

Prototyping with the Entity-relationship Model 40

Rapid Prototyping of Man-machine Interfaces for Telecommunications Equipment Using  
Interactive Animated Computer Graphics 41

Research Aspects of Rapid Prototyping 48

Software Development by Functional Language Prototyping 25

1988

Dictionary-supported Prototyping of Database Applications 53

Distributed Interactive Scenario Generator for Command, Control And Communications, A  
52

Experience with the Management of Software Projects 1988 51

Integrating the Role of Rapid Prototyping and Requirements Specification Using the  
Object-oriented Paradigm 41

Methodical and Management Experiences from an Extensive Software Project 43

Prototyping Environment for Distributed Database Systems, A 60

1989

Applications and Automatic Generation of Intelligent Graphical Interfaces 58

Computer Aided Prototyping Methodology, A 46

Emergence of Rapid Prototyping as a Real-time Software Development Tool, The 31, 42

Ensemble: A Graphical User Interface Development System for the Design and  
Use of Interactive Toolkits 54

Graphical Specification of User Interfaces with Behavior Abstraction 32

IEE Colloquium on 'Specification of Complex Systems' (Digest No.145) 42

Information Systems Design and Prototyping Using an Object-oriented  
Software Engineering Environment 27

Knowledge Base Approach to the Specification Of Real Time System Requirements, A 28

Knowledge-based System for Rapid Prototyping 62

Lessons Learned from the Use of a Spiral Model for an Ada Development Effort  
the Software Life Cycle Support Environment (SLCSE) 25

Multimedia Rapid Prototyping Tool for the Development Of Computer-assisted  
Instruction, A 28

OBSERV: A Prototyping Language and Environment Combining Object Oriented  
Approach, State Machines and Logic Programming 65

Protolog: A Conceptual Schema Facility for Automated Prototype Generation 38

Prototyping and Simulating Parallel, Distributed Computations with VISA 33

Prototyping Techniques for Different Problem Contexts 39

Rapid Prototyping Capabilities in the Expert Requirements Expression and System  
Synthesis (EXPRESS) Environment 51

- Rapid Prototyping for Healthcare Applications 49
- Rapid Prototyping from Structured Analysis: Executable Specification Approach 46
- Rapid Prototyping in Command and Control System Development 65
- Rapid Prototyping of Complex Avionics Systems 27
- Rapid Prototyping of Formal Specifications Using Miranda 53
- Rapid Prototyping of Software Systems Using Prolog 35
- Rapid Prototyping of the Operational Definition of Command and Control Consoles 64
- Reliable, Reusable Ada Components for Constructing Large, Distributed Multi-task  
Networks: Network Architecture Services (NAS) 56
- Software Architecture Modelling 52
- Software Evolution through Rapid Prototyping 48
- Software Quality via Rapid Prototyping 43
- Statemaster: A UIMS Based on Statecharts for Prototyping and Target Implementation 66
- 1990
- AI Techniques and Object-oriented Technology for VLSI Design-space Representation,  
Optimization and Management 41
- Between Man and Machine 31
- C3I: A Neural ASIC for Real-time Prototyping 65
- CHRIS Consultant- A Tool for Database Design and Rapid Prototyping, The 64
- DETAIL: An Approach to Task Analysis 34
- Display Rapid Prototyping and Simulation System, A 38
- DPSOI: An Executable Requirements Specification Language for  
Information Processing Systems 61
- ES-Kit: Rapid Prototyping of Scalable High Performance Systems 58
- Graphical Support for Reducing Information Overload in Rapid Prototyping 48
- Graphical Tool for Computer-aided Prototyping 49
- HyperCard- The Legend: Summary 43
- Impact of DoD-Std-2167A on Iterative Design Methodologies: Help or Hinder?, The 54
- Integrated Approach to the Design and Performance Evaluation of Distributed Systems,  
An 25
- Knowledge-based Approach to Rapid Prototyping Systems, A 63
- Knowledge-based Software in a Realtime Alarm-handling System 36
- Modeller's Assistant- A First Step Towards Integration of Knowledge Bases and  
Modelling Systems, The 45
- Object-oriented Application Development Environment, An 36
- Object-oriented Development Aids Prototyping and Delivery 24
- Pantheon: Rapid Prototyping of Natural Language Interfaces to Large Databases 50
- Prototyping and Visualisation in Interface Design 42
- QUISAP: An Environment for Rapid Prototyping of Real-time Systems 35
- Rapid Ada Prototyping: Principles and Example of a Complex Application 44
- Rapid Prototyping Approach to Software Development and our Tool for Development of  
RPT Prototypes, A 56
- Rapid Prototyping in Ada in the Rational Environment Emphasizing Software Reuse 47
- Rapid Prototyping of Communications Protocols Using a New Parallel Language 30
- Rapid Prototyping of Large Command, Control, Communications and  
Intelligence (C3I) Systems 60
- Rapid Prototyping through Graphical Operational Specification and Automated  
Code Generation 68
- Rapid Prototyping: A Cure for Software Crisis 59

Rapid Prototyping: A Practitioner's Viewpoint in Software Development 59  
Real-time Distributed Simulation of Pbx with Software Reuse 30  
Retrieval of Consistent Documentation for Rapid Prototyping Systems and  
Software Maintenance 63  
RSF: A Formalism for Executable Requirement Specifications 33  
Run-time Environment and Application Tools for the Ground Test Accelerator 45  
SEKE'90. Proceedings. Software Engineering and Knowledge Engineering. 2nd  
International Conference 57  
Survey of Rapid Prototyping Tools 29  
System and Software Requirements Engineering 61  
Tiny Tim-a Smalltalk Toolbox for Rapid Prototyping and Animation of Models 46  
Using Data-flow Description Supported by the Rapid Prototyping Tool for 55

1991

Acquisition Model for the Capture and Management of Requirements for  
Battlefield Software Systems 29  
CAPS as a Requirements Engineering Tool 49  
Durra: An Integrated Approach to Software Specification, Modeling and Rapid Prototyping  
26  
Environment for Rapid Prototyping of Interactive Systems, An 67  
Language Aspects of ENVISAGER: an Object-oriented Environment for the Specification  
of Real-time Systems 34  
Methodology for Prototyping-in-the-Large, A 55  
Rapid Development Speeds Path for Command System 37  
Rapid Prototyping in Human-Computer Interface Development 40  
Simulation of Modern Electronic Combat Scenarios by Means of a Flexible Generic  
Computer Model 66  
Survey of Exploratory Software Development, A 62  
TXL:A Rapid Prototyping System for Programming Language Dialects 32

## Additional References

- [Basili81] Basili, V., and Weiss, D., "Evaluation of a Software Requirements Documents by Analysis of Change Data," *Proceedings of the 5th ICSE*, 1981.
- [Bennett89] Bennett, M.J., "Modeling Radar Countermeasure Systems," *Defense Computing*, July-August, 1989.
- [Boehm75] Boehm, B. W., et al., "Some Experience with Automated Aids to the Design of Large-Scale Reliable Software," *IEEE Transactions on Software Engineering*, vol 1, no 1, March 1975.
- [Boehm88] Boehm, B.W., "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, p. 61, May 1988.
- [CONG90] "Bugs in the Program: Problems in Federal Government Computer Software Development and Regulation," Staff Study by the Subcommittee on Investigations and Oversight, One Hundred First Congress, April 1990.
- [Davis90] Davis, A., *Software Requirements: Analysis and Specification*, Prentice-Hall, 1990.
- [DOD91] "Software Technology Plan: Vol. II Plan of Action," Draft 5, August 15, 1991.
- [DSB87] "Report of the Defense Science Board Task Force on Military Software," Office of the Under Secretary of Defense for Acquisition, U.S. DoD, September 1987.
- [Graham89] Graham, D.R., "Incremental Development: Review of Nonmonolithic Life-Cycle Development Models," *Information and Software Technology*, 31, 1, pp. 7-20, Jan 1989.
- [Humphrey89] Humphrey, W.S., *Managing the Software Process*, Addison-Wesley, 1989.
- [Lientz80] Lientz, B., and E. Swenson, *Software Maintenance Management*, Addison-Wesley, 1980.
- [NOSC90] "Models of Software Evolution: Life Cycle and Process," Naval Ocean Systems Center, NOSC-TD-1893, July 1990.
- [Royce70] Royce, W., "Managing the Development of Large Software Systems," *IEEE WESCON*, August 1970, pp. 1-9.
- [SEI91] *Proceedings of the Requirements Engineering and Analysis Workshop*, Software Engineering Institute, March 1991.
- [Sage90] Sage, A.P., and J.D. Palmer, *Software Systems Engineering*, John Wiley and Sons, 1990.
- [Tavolato84] Tavolato, P., K. Vincena, "A Prototyping Method and Its Tool," *Approaches to Prototyping*, R. Budde et al., eds., Berlin: Springer-Verlag, 1984. pp. 434-446.

## REPORT DOCUMENTATION PAGE

<b>1a. REPORT SECURITY CLASSIFICATION</b> Unclassified		<b>1b. RESTRICTIVE MARKINGS</b> None																
<b>2a. SECURITY CLASSIFICATION AUTHORITY</b> N/A		<b>3. DISTRIBUTION/AVAILABILITY OF REPORT</b> Approved for Public Release Distribution Unlimited																
<b>2b. DECLASSIFICATION/DOWNGRADING SCHEDULE</b> N/A																		
<b>4. PERFORMING ORGANIZATION REPORT NUMBER(S)</b> CMU/SEI-92-TR-13		<b>5. MONITORING ORGANIZATION REPORT NUMBER(S)</b> ESC-TR-92-013																
<b>6a. NAME OF PERFORMING ORGANIZATION</b> Software Engineering Institute	<b>6b. OFFICE SYMBOL (if applicable)</b> SEI	<b>7a. NAME OF MONITORING ORGANIZATION</b> SEI Joint Program Office																
<b>6c. ADDRESS (City, State and ZIP Code)</b> Carnegie Mellon University Pittsburgh PA 15213		<b>7b. ADDRESS (City, State and ZIP Code)</b> ESC/AVS Hanscom Air Force Base, MA 01731																
<b>8a. NAME OFFUNDING/SPONSORING ORGANIZATION</b> SEI Joint Program Office	<b>8b. OFFICE SYMBOL (if applicable)</b> ESC/AVS	<b>9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER</b> F1962890C0003																
<b>8c. ADDRESS (City, State and ZIP Code)</b> Carnegie Mellon University Pittsburgh PA 15213		<b>10. SOURCE OF FUNDING NOS.</b> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <tr> <td style="width: 25%;">PROGRAM ELEMENT NO</td> <td style="width: 25%;">PROJECT NO.</td> <td style="width: 25%;">TASK NO</td> <td style="width: 25%;">WORK UNIT NO.</td> </tr> <tr> <td>63756E</td> <td>N/A</td> <td>N/A</td> <td>N/A</td> </tr> </table>		PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT NO.	63756E	N/A	N/A	N/A							
PROGRAM ELEMENT NO	PROJECT NO.	TASK NO	WORK UNIT NO.															
63756E	N/A	N/A	N/A															
<b>11. TITLE (Include Security Classification)</b> A Classification and Bibliography of Software Prototyping																		
<b>12. PERSONAL AUTHOR(S)</b> David P. Wood and Kyo C. Kang																		
<b>13a. TYPE OF REPORT</b> Final	<b>13b. TIME COVERED</b> FROM                      TO	<b>14. DATE OF REPORT (Yr, Mo., Day)</b> October 1992	<b>15. PAGE COUNT</b> 90															
<b>16. SUPPLEMENTARY NOTATION</b>																		
<b>17. COSATI CODES</b> <table border="1" style="width: 100%; border-collapse: collapse; margin-top: 5px;"> <thead> <tr> <th style="width: 33%;">FIELD</th> <th style="width: 33%;">GROUP</th> <th style="width: 33%;">SUB. GR.</th> </tr> </thead> <tbody> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> <tr><td> </td><td> </td><td> </td></tr> </tbody> </table>		FIELD	GROUP	SUB. GR.													<b>18. SUBJECT TERMS (Continue on reverse of necessary and identify by block number)</b> bibliography software process software prototyping	
FIELD	GROUP	SUB. GR.																
<b>19. ABSTRACT (Continue on reverse if necessary and identify by block number)</b> <p>Prototyping, the creation and enaction of models based on operational scenarios, has been advocated as a useful software engineering paradigm because it lends itself to intense interaction between customers, users, and developers, resulting in early validation of specifications and designs. An extensive and widespread interest in software prototyping in recent years has resulted in a daunting amount of literature and dozens of proposed methods and tools. As with any immature and growing technology, the expanding literature and approaches have resulted in correspondingly expansive and confusing terminology.</p> <p>This report presents an overview of technology and literature relating to the creation and use of software system prototypes. In addition to an annotated bibliography of recent prototyping literature, a technology framework, taxonomy, and series of classifications are provided. The intent of this report is to provide a basic road map through the available literature and technology.</p> <p style="text-align: right;">(please turn over)</p>																		
<b>20. DISTRIBUTION/AVAILABILITY OF ABSTRACT</b> UNCLASSIFIED/UNLIMITED SAME AS RPTDTIC USERS		<b>21. ABSTRACT SECURITY CLASSIFICATION</b> Unclassified, Unlimited Distribution																
<b>22a. NAME OF RESPONSIBLE INDIVIDUAL</b> Thomas R. Miller, Lt Col, USAF		<b>22b. TELEPHONE NUMBER (Include Area Code)</b> (412) 268-7631	<b>22c. OFFICE SYMBOL</b> ESC/AVS (SEI)															

ABSTRACT —continued from page one, block 19