# PROTOTYPE DEVELOPMENT : A SOFTWARE ENGINEERING ISSUE†

## 8.1. Introduction

Expert system development is the first and foremost Software engineering [1]. This is reflected in the importance of such issues as portability, integration, data base access, fielding, maintainability, robustness, reliability, concurrent access, performance, user interface, debugging support, and documentation. Two important contributions of ES-technology might be pointed out here (i) it has shown how to encode knowledge explicity and declaratively rather than implicity and procedurally, and thereby making programs more understandable and maintainable; and (ii) it is the pioneer of the new software development strategy of prototyping and refinement as opposed to the classical phase refinement approach.

Perhaps the most difficult part of constructing a large software system is deciding exactly what to construct. Requirements analysis, as software engineers call it, is the most crucial phase in the life of a project. An error in this phase can not only add to the time required for completing the project, but also lead to a delayed product that was not required in the first place [2].

The difficulty arises from two sources. Typically, the would-be users do not quite know what they want, at least not until they have tried out some version of the program. Compounding this difficulty is the fact that the planner of any software design activity does not have an exhaustive repertoire of questions that will yield him the necessary information. Neither can he be sure that the specifications he has derived are complete; chances are, they are not. Rapid prototyping has been touted as a way of this deadlock [3].

In the next section we have discussed the concepts of prototying and prototyping cycle. In section 8.3, a comparison has been made for phase refinement with prototyping approach. In section 8.4, the evolution stages of an expert system have been discussed. Section 8.5 presents our prototype 1.0. In section 8.6, our conclusions and discussion are summarized.

---

† This is based on the publications [ Proc. Nat. Conf. on Software Engineering and its Application, Hyderabad, India, August 23-25, 1996, 127-134; CSI Communications, May 1997,15-21, ibid June 1997, 21-25 ] of the author.

## 8.2. **Prototyping and prototyping cycle**

Prototyping is the process of developing a scaled-down version of a system to use in building a full-scale system. The primary purpose of prototyping is to reduce time and expense in building quality systems. Prototyping mandates a philosophy of incremental system development that includes end users in the assessment of emerging system capabilities. Increased participation of end users leads to faster system development and ultimately to more useful systems [4]. Although recent innovations in prototyping [4-6], suggests the importance of end-user involvement, they do not go far enough to bring the end user into software development.

In contrast to traditional prototyping, knowledge engineering techniques commonly employ domain experts (experts in the problem area) as representative end users [7]. Prototype development in knowledge engineering sessions progresses through incremental refinement of a domain model, typically extending over two or more years.

Fig. 8.1 illustrates the iterative prototyping cycle [8]. The user and the designer work together to define the requirements and specifications for the critical parts of the envisioned system. The designer then constructs a model or prototype of the system in a prototype description language at the specification level. The resulting prototype is a partial representation of the system, including only those attributes necessary for meeting the requirements. It serves as an aid in analysis and design rather than as production software.

During demonstrations of the prototype, the user evaluates the prototype's actual behavior against its expected behavior. If the prototype fails to execute properly, the user identifies problems and works with the designer to redefine the requirements. This process continues until the user determines that the prototype successfully captures the critical aspects of the envisioned system.

The designer uses the validated requirements as basis for designing the production software. Additional work is often needed to construct a production version of the system. For example, the prototype :

a) might not include all aspects of the intended system,

b) might have been implemented using resources that will not be available in the actual operating environment,

c) might not be able to handle the full workload of the intended system, or

d) might meet its timing constraints only with respect to linearly scaled simulated time.

Experience with production use of a delivered system often leads to new customer goals, triggering further iterations of the prototyping cycle.

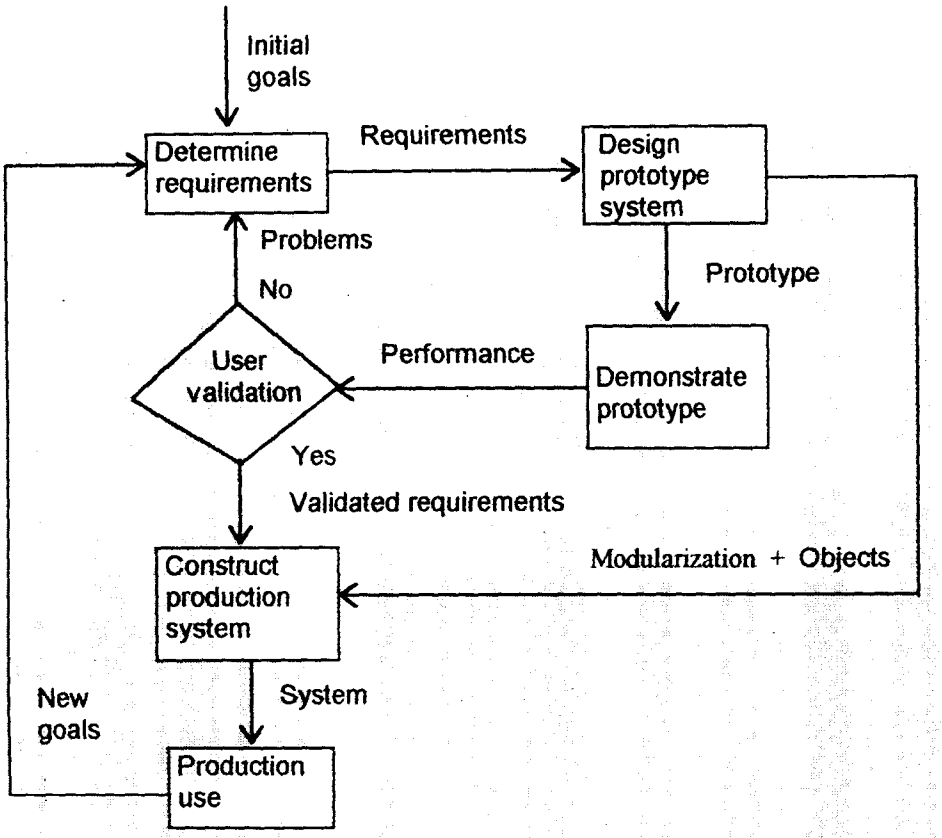


Fig. 8.1 The prototyping cycle

## 8.3. **Phase refinement vs. prototyping**

The classical phase refinement approach of system design is criticized for its high expense and long time for development of quality systems. Prototyping has been proposed for system development which reduce time and cost substantially. Prototyping is a paradigm which consists of some non-standard concepts and suggests increased participation of end users. Prototypes are subject to frequent and repeated changes to incorporate the suggestions from the human experts as well as from the end users. To find the benefits of prototyping than a conventional approach one has to apply : computer aided prototyping and object-oriented prototyping [8]. The first one provides mechanical assistance and the second one provides conceptual simplicity. To make the prototypes more flexible as well as achieve automation easier. O-O approach may be suitable.

While the above analysis might lead one to believe that the prototyping paradigm has all the good things in connection with the development of quality systems, the paradigm does have some bottlenecks. After evaluating the software request one has to determine whether the software to be developed is a good candidate for prototyping. Not all software is amenable to prototyping. Owing to poor project discipline, prototyping may increase cost and time. A number of prototyping candidacy factors [9] can be defined : application area and complexity, customer characteristics and project characteristics. In general, any application that creates dynamic visual displays, interacts heavily with a human or demands combinatorial processing that must be developed in an evolutionary fashion is a candidate for prototyping [10].

For a problem domain of large and varied knowledge base, an expert system should be developed in an evolutionary framework. The designers usually follow five stages as identified by Waterman [11] (section 8.4). Prototyping may have an adverse impact on modifiability and maintainability of knowledge bases since these may be patched and modified several times during the process of evolution of an expert system to get it in its commercial form. This may, however, be overcome by the use of O-O approach. As the system grows, the major changes will be with the addition of new and deletion of old objects rather than modifying the old objects. In this respect O-O approach is considered very useful for rapid prototyping.

## 8.4. **Stages of expert system evolution** [11]

Waterman identifies the following **five** stages in the evolution of an expert system :

| Development Stage | Description |
|---|---|
| Demonstration prototype | The system solves a portion of the problem undertaken, suggesting that the approach is viable and system development is achievable. |
| Research prototype | The system displays credible performance on the entire problem but may be fragile due to incomplete testing and revision. |
| Field prototype | The system displays good performance with adequate reliability and has been revised based on extensive testing in the user environment. |
| Production model | The system exhibits high quality, reliable, fast and efficient performance in the user environment. |
| Commercial system | The system is a production model being used on a regular commercial basis. |

## 8.5. Prototype 1.0

We now present our first demonstration prototype. During the total system development, we have taken care of the "five-stage" development suggestions of Buchanan et. al. [5]. Konwledge was acquired from the source shown in **Appendix A**.

### 8.5.1. General description

The architectural components of our system is shown in fig. 8.2. There are two key aspects involved in the process of design an expert system : Knowledge Base(s) and Inference Engine. The knowledge gathered from domain experts is stored in knowledge base (KB). The KB consists of two parts : static part and dynamic part. The static part is relatively fixed over time. The dynamic part is capable of adding new facts or facts can be removed from the KB as when required. After entering the relevant knowledge, one can save the KB(s) in the external storage for later use.

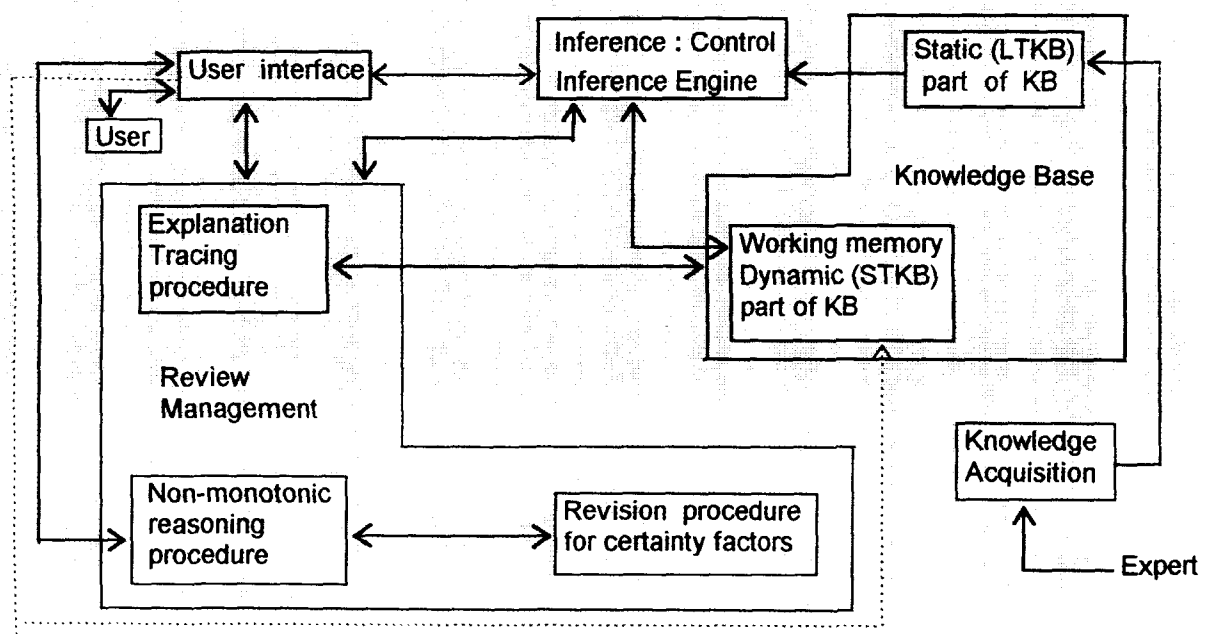


Fig. 8.2. Architectural components of the system

The inference engine uses LTKB and STKB to infer new facts. Backward reasoning process has been used here which favours the needs of the application domain. It has been shown below that the inference engine uses depth-first scanning but with an 'improved back tracking'.

A user interacts with the system with the user interface of the system. Through this module different queries are served by the system initiated by the inference engine.

The total review management is transparent to the user through this particular module. All accesses by a user to KB and review management module are through inference engine. However, logical access is presented through broken line of fig. 8.2. Knowledge acquisition module is responsible for enhancing the system knowledge by the knowledge engineer as when acquired from domain experts.

A more detail discussion on the organization of the knowledge base, the inference engine and review management is provided in the following sub-sections.

### 8.5.2. **Knowledge Base**

The domain experts provide the knowledge of the problem area. After extracting the necessary knowledge from the domain experts, one has to implement that knowledge in a correct and efficient knowledge base. It is the most important part of the expert system which is made separated from the inference engine with some well-known advantages. The knowledge base consists of two parts : static part and dynamic part. The static part of the knowledge base does not change over time on long term basis which may be termed as long-term knowledge base (LTKB). The dynamic part of the knowledge base is used from where facts can be added to or facts can be removed from the knowledge base as when required. This dynamic part is used on short-term basis which may be termed as short-term knowledge base (STKB). Facts obtained from the user during consultation session that apply only to the current consultation are stored in this dynamic part. This STKB may act as a part of the nonmonotonic reasoning process. This STKB also helps in achieving 'improved backtracking' which will be explained in sub-section 8.5.3. The dynamic knowledge base is stored in memory with the static knowledge base. One can save the dynamic as well as static knowledge base in a secondary storage device for later use.

### 8.5.3. **Inference Engine**

The inference engine has two functions : inference and control. Inference is the basic formal reasoning process which involves matching and unification. Such inference operates by modus-ponens. The control function determines the order in which the rules are tested and what happens when a rule succeeds or fails. The control function must also handle the well-known problem of conflict resolution. The inference engine scans the rules using backward chaining during the consultation process; i.e. starts at the goal and works backward. In this type of application of medical diagnosis, backward chaining is useful because the questioning is guranteed to follow the focused goal conclusion. The scanning used in our application is limited to depth-first nature. As a general principle, all rules relative to a particular goal are scanned as deeply as possible for a solution before it backtracks and tries an alternative goal. However, we may control that backtracking using STKB and using

some control rules. The STKB also helps in eliminating repetitive questioning during consultation session. This results in considerable speed up during inferencing. For the purpose, in the design we have added some control aspects using some control rules that are not a part of the knowledge base proper. The static knowledge base, then really contains two types of rules : knowledge base rules and control rules. Using some control structures we may eliminate certain search paths in problem space resulting in 'improved backtracking' provided that sufficient domain knowledge is available. The overhead of conventional chronological backtracking where no prior knowledge about the prospective backtrack point is available, can be much reduced using 'improved backtracking'.

### 8.5.4. **Review Management**

A review on the conclusions can be made during the consultation sessions. Under the general heading of review management, there are mainly three procedures involved, namely, Explanation tracing procedure, Nonmonotonic reasoning procedure and Certainty factor revision procedure. The Explanation tracing procedure provides explanations to mainly 'How' a conclusion has been achieved. The system may provide a sequence of 'fired rules' to achieve a particular goal. ' WHAT IF ' type of review is provided with a combination of nonmonotonic reasoning and certainty factor revision procedure. ' WHAT IF ' review can be used to find out WHAT conclusion will be deduced IF certain certainty factors are changed.

### 8.5.5. **Implementation**

For the present implementation we have taken the language Turbo Prolog. The basic reasons behind the choice are as follows: (1) It has its fast prototyping capability of developing expert systems [12], (2) In-built inference mechanism which expedites initial prototype development of the system, (3) Separation between knowledge base and inference mechanism, (4) Static and dynamic knowledge bases: the dynamic portion of the knowlwdge base remembers the results of the previous consulting session which may be considered as an intelligent activity of the system. In addition, facts can be added to or can be removed from the dynamic portion of the knowledge base which is one of the demanding characteristics of nonmonotonic reasoning, (5) Declarative as well as some procedural knowledge can be intermixed: this is a demanding characteristic of the object oriented approach and which also resolves declarative-procedural knowledge controversy in a restricted manner, (6) Supports modular and structured programming to achieve good modifiability and to satisfy the demand of the object oriented approach, (7) Table look-up scheme for fuzzification and the defuzzification processes can be easily implemented using TURBO-PROLOG's list-structure, and (8) The backward chaining favours the medical diagnosis problems.

### 8.5.6. **Analysing Process**

To initiate its analysis activity, an expert system requires some initial information about the application domain. The system applies rules on these facts and new facts are generated. Recursive application of this process leads to the goal, if satisfied. Dynamically generated new facts / results are stored in STKB for comparison, explained later on. In the process, different certainty factors of premises are supplied in actual runs. The system will ask for these factors and we should supply these from parents' informations about the child. It will calculate the certainty factors of the inferences of its own from the supplied factors of the premises. For a typical analysis, we have considered those activities as "highly likely abnormal" whose certainty factors are less than 0.6. To compare the performance of the child's current data with the previous consultation date, we preserve facts / results of one previous session in STKB. During the current session we compare the current CFs with the previous CFs. It calculates the percentage of increase or decrease or no change of performance of a child. This activity comparison is essential to comment on the overall performance profile of a child. For a typical conclusion, it calculates the overall performance and will display a message like "The growth has decreased in 45% cases". The variation in activity by 10% has been considered negligible. Then it draws a general conclusion like "The growth of the baby is expected to be abnormal". It is essential then to know the activites which are abnormal for the child. The system displays those abnormal activities which should get special attention for treatment. In addition, user may want to see the tracing of reasoning to goal for his / her satisfaction. The system may display the firing rules with their corresponding CFs. This tracing of reasoning to goal also helps a medical professional to get an idea about the sequence of firing rules with CFs for further actions. If one is not satisfied with the conclusion, he / she may want to change the CF of a rule at this stage. The supply of subjective informations by the parents / guardians of a child needs this revision of certainty factors. The system accommodates this revision and a fresh conclusion will be drawn. If the user is fully satisfied with the conclusion drawn, it will move to the next session, if desired by the user.

### 8.5.7. **Consulting the system**

We now present an excerpt from a typical consultation session with the system.

```
                    WELCOME TO
THE CONSULTATION SYSTEM FOR CHILD GROWTH AND DEVELOPMENT

Reference Number    : r3 - 6 - 13
Name of the baby    : Jai
Father's Name       : Dhajen
Mother's Name       : Mira
Address             : Shibmandir
Age                 : One Month

                                Press any key to continue ...
```

Do you want to compare certainty factors (y/n) ? y

| Certainty Factors | | | |
|---|---|---|---|
| Activity | Current value on 13/6/1994 | Previous value on 12/5/1994 | Comments |
| Axial muscle tone | 0.3 | 0.8 | Decreased by 50% |
| Muscle tone of limbs | 0.6 | 0.9 | Decreased by 30% |
| Spontaneous gestures | 0.6 | 0.8 | Decreased by 20% |
| Visual | 0.6 | 0.9 | Decreased by 30% |
| Gripping | - | - | No change |
| Relation | 0.7 | 0.8 | Decreased by 10% |
| Activity | 0.79 | 0.8 | Decreased by 1% |
| Language | - | - | No change |
| Rhythem | 0.7 | 0.8 | Decreased by 10% |
| EEG | 0.99 | 0.9 | Increased by 9% |
| | | | Press any key to continue ... |

Do you want to know the general conclusion (y/n) ? y

A CONSULTATION SYSTEM FOR CHILD GROWTH AND DEVELOPMENT

Growth of the baby decreasing in 45.45% cases.
The variation upto 10% has been neglected.

CONCLUSION

The growth of the baby is expected to be abnormal.
The activities which are most abnormal for the child:
Axial_Muscle_Tone

Press any key to continue ...

Do you want to trace of reasoning to goal (y/n)? y

A CONSULTATION SYSTEM FOR CHILD GROWTH AND DEVELOMENT

Following sequence of rules leads to conclusion :
query(group_1, one)
activity(axial_ muscle_ tone,one)
by firing rule 'minm' :
Mininum value is : 0.3
i.e. CF of the above rule is : 0.3
activity(muscle_ tone_ of_ limbs, one)
by firing rule 'minm' :
Minimum value is : 0. 6
by firing rule 'minm1' :
Minimum value is : 0. 6
by firing rule 'minm2' :
Minimum value is : 0. 6
by firing rule 'minm3' :
Minimum value is : 0. 6
i.e. CF of the above rule is : 0. 6
activity(spontaneous_ gestures, one)
by firing rule 'minm' :
Minimum value is : 0. 8

Press any key to see next ...

:
:

A CONSULTATION SYSTEM FOR CHILD GROWTH AND DEVELOPMENT
:
:

By firing 'comparing - factor' it concludes :
Growth of the baby decreasing in 45.45% cases

Press any key to continue ...

Hope you are satisfied (y/n) ? n
Do you want to change current value of any certainty factor
(y/n) ? y
To change the CF of an activity, give the activity no : 3
Enter new CF : 0.8
Do you want to change more (y/n) ? n

```
A CONSULTATION SYSTEM FOR CHILD GROWTH AND DEVELOPMENT

The current data has been stored
Please wait for further information ...

        Growth decreasing in 36.36% cases
        The variation upto 10% has been neglected

                C O N C L U S I O N

The growth of the baby is expected to be abnormal
The activities which are abnormal for the child :
Axial_Muscle_Tone

                                        Press any key to continue ...
```

Do you want to trace of reasoning to goal (y/n) ? n
Hope you are satisfied (y/n) ? y
Do you want to consult further (y/n) ? n
Good bye friend ! See you !

The above excerpt clearly demonstrates how you can interact with the system. The reply from the system is displayed through screen-based frames for compactness of relevent informations.

## 8.6. Conclusions and Discussion

This chapter provides a prototype of our developed system. A small review of prototyping and prototyping cycles have been provided. A comparison has been made between phase refinement and prototyping. Five stages in the evolution of an expert system as identified by Waterman have been discussed. The architecture of the system have been presented and discussed. Typical consultation sessions have been presented.

Uncertainty in informations has been considered in terms of certainty factors in MYCIN style. Other forms of inexactness ( e.g., fuzzy informations ) will be presented in the later parts of the thesis. The present system contains static as well as dynamic databases which offers an additional intelligent activity of the system. The system is also portable and can be run on IBM-compatable systems. Last of all, we must mention that the system in its present form may not be used in a real medical diagnosis which needs more standarization. The standarization may be achieved through users' participation and through the knowledge acquisition of different experts in the

domain. The issue of system validation and testing has been provided at the later stages of our development.

**References**


1. J. Rothenberg. Expert system tool evaluation. Topics in expert system design, G. Guida and C. Tasso, eds., Elsevier Science Publishers B. V. ;(North-Holland), 1989, 228.

2. F. P. Brooks. No silver bullet. IEEE Computer; vol. 20, no. 4, April 1987, 10-19.

3. R. S. Pressman. Software Engineering. McGraw-Hill; New York, 1987.

4. Bernard H. Boar. Application prototyping : A requirements definition strategy for the 80's. John Wiley and Sons; New York, 1984.

5. B. G. Buchanan et. al. Constructing an Expert System. In Building Expert Systems, Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. (eds.), Addison-Wesley, Reading, Mass.; 1983, 127-167.

6. Barry W. Boehm. A spiral model of software development and enhancement. IEEE Computer; vol.21, no.5, May 1988, 61-72.

7. Robert R. Hoffman. The problem of extracting the knowledge of experts from the perspective of experimental Psychology. AI Magazine; vol.8, no.2, Summer 1987, 53-67.

8. Luqi. Software evolution through rapid prototyping. IEEE Computer; vol.22, no.5, 1989,13-25.

9. B. Boar. Applications Prototyping. Wiley-Inter Science; 1989.

10. Roger S. Pressman. Software Engineering - A Practitioner's Approach. (3rd ed.), McGraw-Hill, Inc.; 1992.

11. Donald A. Waterman. A guide to expert systems. Addison-Wesley, Reading, Mass.;1986, 136-141.

12. Ivan Bratko. Fast prototyping of expert systems using prolog. Topics in expert system design, G. Guida and C. Tasso, eds., Elsevier Science Publishers B. V.; (North-Holland), 1989.