1. Introduction

Critical to the success of any engineering or construction project is the prior identification, specification and validation of the requirements that are to be satisfied by the intended product. Specification presupposes preciseness, preciseness presupposes mathematical models. Unfortunately, such models do not always exist in real world practice, or, if they do exist, often are not comprehensive enough. In order to overcome these shortcomings, it has become good practice in all engineering disciplines to work with substitutes like lab models or simulation models. Such a substitute is called a prototype if it exhibits a substantial part of the functional properties of the final product.

Software construction is an engineering discipline. In fact it is one where the lack of exact models and techniques is pronounced. In addition, there is practically no appreciable knowledge base common to all of the three groups: theoreticians, system analysts and users. This enormously handicaps both, the introduction of formal approaches into the application environment and the communication of practical needs to the theoretician.

Again, prototyping shows a way out of this dilemma [SMITH, SQUIRES, MAYR, MASON]: Used as a live communications vehicle, software prototypes are much more understandable to the potential user than any written piece of paper. Used for experimenting during requirement analysis, software prototypes may help to extract definitive requirements from vague ideas, and to validate these requirements (functional and/or...
behavioural properties, interface etc.) in close co-operation between analyst and customer. In summary, software prototypes support the PARTICIPATIVE APPROACH [MUMFORD] to system design.

One can hardly expect that there is one universal approach to prototyping independent of the intended application. Instead, different software classes need different prototyping approaches and tools. What we plan to study in this paper is prototyping of interactive application systems (IAS), i.e. software systems that interactively solve clerical, managerial and expert problems within business and technical environments.

It will be shown that prototyping is a rather natural approach in this context since it is no more than a slight extension of modern IAS architectural concepts and construction techniques.

In detail, section 2 points out some general requirements of software prototyping and relates them to concepts of application system engineering. Section 3 discusses matters in more detail by presenting a concept for highly adaptable IAS user interfaces that immediately leads to an add-on IAS architecture as described in section 4. In section 5 we revisit IAS prototyping by relating it to the concepts developed up to that point. Finally, we suggest future developments and further research goals.

2. IAS Prototyping: Premises

Let us start with some general remarks on how software prototyping may and should be done, and by which tools it can and should be supported. One important demand is an immediate consequence of the experimentation aspect of prototyping: experimenting is only worthwhile if it may be performed with changing scenarios. As a consequence, prototypes must be quickly modifiable and again available for further investigation. Hence we need some kind of PROTOTYPER'S WORKBENCH into which we may chuck a workpiece prototype for manipulations and controlled running.

A prototype and the information gained from it are an integral part of the results obtained for the intended software product during the early phases of its life cycle. Furthermore it is conceivable that parts of a prototype may be used to derive the functional specification for the corresponding component of the final system. Provided certain quality standards have been met, some prototype software may even become part of the final system itself. Hence, as Figure 1 points out, immediate results of prototyping may show up in all phases of the life cycle. In the extreme, the prototype may itself be the final product. These reasons make it obvious that the tools and techniques used for prototyping should be fully integrated into the software production environment one is actually working with.