Vision Statement for Software Architecture Support System (SASSY)

The goal is to produce a software system that assists with the task of creating an architecture for a software system.

The current process involves writing text documents, typically using a word processor, and trying to embed into them a set of diagrams that describe the proposed design at a high level. The results are generally not very satisfactory for a variety of reasons.

Creating diagrams for inclusion into word processing documents is not easy. Some word processors are quite poor at handling diagrams. I have seen MS Word suddenly decide that a diagram should be moved to the footer several pages ahead. There are serious limits to the complexity that can be described in such a diagram. For a complex system the diagram will either be too cluttered to be readable or such a high level view that it is essentially meaningless.

Creating diagrams is often manually intensive. Even with diagramming editors that have the appropriate symbols and an understanding of what the symbols represent it is still necessary to place the symbols and label them. This takes a considerable amount of tedious effort. The result is that diagrams can quickly become out of date as the design evolves but there is not time to revisit the diagrams. There is also a tendency to combine several aspects of the design into the one diagram. This can make the diagrams confusing and ambiguous.

The use of word processing documents to describe the system also has its problems. When writing a document it is usual to target a specific audience - perhaps the developers, perhaps the stakeholders, for example. It is very difficult to write a single document for the use of a variety of audiences. It will have too much detail that is not relevant for most of its readers, and conversely each reader will have difficulty finding the information that they need.

The rather obvious answer is to create numerous documents and diagrams each tailored to a specific aspect of the design. However this brings with it another problem, namely, how to keep a large collection of objects consistent while the design goes through its inevitable evolution?

The current process for developing a software architecture does not scale very well. For very large projects it becomes necessary to use a significant number of 3rd party products (unless you have an extraordinarily large budget). This range of products can easily exceed the knowledge of any single architect. Moving to a team of architects presents its own problems, however, since the project will no longer have a single visionary to guide it.

The objective of this project is to solve these problems. The approach is to capture the design, the software architecture, in an ontology or knowledge database. Being in a single repository it can be kept consistent, and in fact the tools available for checking ontologies can be used to identify any inconsistencies. We will then develop a set of tools that can generate the documentation and diagrams for each aspect or view that is required. A single repository, with multi-user access, can allow multiple architects to collaborate on the design process by enabling each one to easily see the information that is relevant to the part of the project that they are working on.

While the primary target is the architecture for a software system, it may be extended to the analysis and detailed design, and may also be applicable to other domains besides software.

There will be an initial Software Architecture Knowledge Base which will capture the essence of software architecture. The user will then use the preliminary analysis, requirements analysis, prototyping, and component exploration results to put specific instances into the knowledge base for their project.

Software will then extract the data from the knowledge base and feed them into programs that will generate the text and diagrams. A user interface component will coordinate the process.

The output format should be PDF so that there is less temptation to edit the output documents rather than the source knowledge base.