Appearing in: The First Working IFIP Conference of Software Architecture, San Antonio, Texas, February 1999.

Preparing for Change in the Architecture Design of Large Software Systems

R. L. Nord, C. Hofmeister, and D. Soni
Siemens Corporate Research
755 College Road East, Princeton, NJ 08540
{rnord, crh, dilip}@scr.siemens.com

Abstract

Architecture is influenced by organizational factors, market forces, and technology that are constantly changing. We found that successful architects analyze factors that have a global influence to produce an architecture that localizes the effects of change. Change is a fundamental property of all factors that influence the architecture. To account for the influence of change we identified a global analysis task that captures the factors an architect considers when designing an architecture. We are describing four industry software systems using this analysis approach.

Keywords

Software engineering, software architecture, industrial applications, engineering concerns.

1 ARCHITECTURE INFLUENCES AND DECISIONS

Software architecture is the structure of the components of a system, their interrelationships, and principles and guidelines governing their design and evaluation over time (Bass et al., 1998). Architecture is influenced by the developing organization, market forces, and technology (Clements, 1995). These factors are constantly changing. We have heard the sentiment expressed that if only the requirements, technology, and the design team were fixed up front, then the product could be built "right." But the reality is that any product built without taking into account that these factors may change, will likely be obsolete the moment it reaches the marketplace.

From our experience of surveying industrial practice in software architecture (Soni et al., 1995), change is a fundamental property of all factors that influence the architecture. In addition to ensuring the architecture can be implemented, the architect must also understand and prepare for how the system will change.

Domain-related product requirements are likely to change during development and over the lifetime of a product. The physical characteristics of the system may

change as new models are introduced. The way users interact with the system is likely to change as they become more sophisticated in using the system and desire more effective ways of using it. As the processing power increases, work that was previously the responsibility of the users shifts to the system.

The technology and requirements affecting the run-time structure of the system are likely to change over the lifetime of the product. It is desirable to port the product to new hardware and that may mean adapting to a new software platform as well. The architecture may change as the system is tuned to meet its performance requirements. The performance characteristics aren't always entirely predictable. Changes in the assignment of components to tasks and task attributes may be needed as the system is tuned.

Aspects of change are typically recorded as a requirement or quality goal. For example, the quality goal "ease of change" is documented and realized in the architecture by information hiding (Parnas, 1972). We soon found from our experience that we needed additional information to capture the rationale for the architecture. We needed a more precise meaning of change. Change could mean adding new features, upgrading hardware, or increasing the data rate specified in a requirement. We needed to express a repertoire of design strategies. Information hiding is one important strategy, but there are additional means to separating software engineering concerns to prepare for future changes. We needed a way to trace the factors to aspects of the architecture at a finer level of granularity.

To account for the influence of change we identified a *global analysis* task that starts as the architecture is defined. Its purpose is to capture the global influencing factors an architect considers and their relationship to strategies that will be used throughout the architecture design. A common set of strategies guiding the developers ensures the integrity of the architecture. This is an iterative process. During the design, certain decisions feed back into the global analysis, resulting in new strategies.

2 GLOBAL ANALYSIS

Global analysis begins with identifying the factors that affect the design of the architecture. These influencing factors fall into three categories: organizational, technological, and product. Table 1 lists typical categories of influencing factors based on our observations. Within each category there will be a number of factors. For example, the schedule (O4) will record the time to market and how features are to be delivered.

Table 1: Example Categories of Influencing Factors

Organization	*Technology*	*Product*
O1: Management	T1: General-purpose Hardware	P1: Features
O2: Staff Skills	T2: Domain-specific Hardware	P2: User Interface
O3: Development Environment	T3: Software Technology	P3: Performance
O4: Schedule	T4: Architecture Technology	P4: Recovery
O5: Budget	T5: Standards	P5: Diagnostics

Organizational factors constrain the design choices. They are external to the product being designed and exert influence on the product mostly during development. Their influence is important because if they are ignored, the architecture may not be buildable. The technological factors are also external to the product being designed. Unlike the organizational factors, however, they can affect the product throughout its lifetime. Further, they can change over time, so the architecture should be designed with this changeability in mind. The product factors, including product requirements, are the primary influence on the architecture. The product factors are also subject to change over time, so the architecture should be designed to support such changes.

The purpose of global analysis is to uncover all of the influencing factors and identify strategies for handling them. The strategies address the global concerns, but provide guidance for implementing them locally. These factors and strategies should be recorded as part of the architecture. To arrive at these strategies we have identified four steps: (1) identify and describe the factors, (2) characterize the changeability of the factors, (3) analyze the impact of the factors, (4) identify the objectives and develop global strategies.

Step 1: Identify and Describe the Factors

At this early stage in the architecture design, the primary factors to consider are those that have a significant global impact, those that could change during development, those that are difficult to satisfy, and those with which there is little experience. To determine whether a factor has a significant global influence, the architect should

ask: can the factor's influence be localized to one component in the design, or must it be distributed across several components; during which stages of development will the factor be important; does the factor require any new expertise or skills?

We illustrate the steps with an excerpt from a data processing system. A probe takes sensor readings that are processed according to the type of acquisition selected by the user. For our example system, we record factors in each of the three categories for illustrative purposes in Table 2. In Step 1, the first two columns are filled in.

Table 2: Characterization of the Influencing Factors

Factor	*Description*	*Changeability*	*Impact*
O4.2 Schedule Feature Delivery	Features are prioritized	Negotiable	Moderate impact on meeting the schedule.
T2.1 Domain-specific Hardware Probe Hardware	This is the hardware to detect and process signals.	Probe hardware upgraded every three years as technology improves.	Large impact on components involved in acquisition and sensor processing.
P1.1 Features Acquisition Types	Acquire raw signal data and convert into sensor readings.	New types of acquisitions may be added every three years.	Affects acquisition performance, sensor processing, and user interface.

The schedule factor shows the features to be delivered are prioritized. For other systems these kinds of factors may not affect the architecture, but in this system they will have a significant impact. For the domain-specific hardware, we have the probe hardware. For product factors, there are a variety of acquisition algorithms to acquire raw signal data and process it into a human readable form.

Step 2: Characterize the Changeability of the Factors

To characterize the changeability of a factor, the architect should ask: how likely will this factor change; will the factor be affected by changes in other factors; how often will the factor change; in what way could it change? These changes could arise either from the volatility of the factor or from the need to be flexible in its usage.

The results of this step are shown in the third column of Table 2. The company's priority is time-to-market. There is flexibility in delaying non-critical features to a later release. The system interacts with the domain-specific hardware of the probe. We want to build flexibility into the system to allow for new models of the probe to be introduced and for different hardware configurations to be specified (e.g., for low-

end to high-end products). Advances in hardware, software, and the application domain will likely drive changes in requirements as technology enables the user to do new things. Acquisition types are expected to be added or enhanced every three years.

Step 3: Analyze the Impact of the Factors

In this step the impact of the factors are analyzed. The impact of factors should be recorded in terms of their impact on particular components; this is not possible if the components have not yet been defined at this point in the architecture design. However, experienced architects will, even at this early stage, have a notion of the high-level components of the system. As the design progresses, the impact of change can be more precisely determined. To determine the impact of change for each factor, the architect should ask, if the identified changes were to happen, would any of the other factors, components, modes of operation of the system, or other design decisions be affected, and if so how?

For the example system the schedule and flexibility of delivery of features have a large and pervasive impact on almost all of the architecture design as noted in the fourth column of Table 2. A change in the probe hardware affects all components involved in acquisition or sensor processing. Adding new features may affect the system's performance, while any changes in the user interaction model will affect the system interface.

Step 4: Identify Objectives and Develop Global Strategies

In this step, the important objectives are identified. These are critical issues that arise from the need to reduce the impact of changeability of factors. For example, one would like to design the architecture to reduce the cost of porting the system to another operating system. Strategies are developed to address identified objectives in order to ensure the implementability and changeability of the design. Table 3 lists some of the strategies we found in the example system.

Table 3: Examples of Strategies

Organization	*Technology*	*Product*
Reuse existing components	Encapsulate hardware	Use feature-based components
Build rather than buy	Separate processing, control, and data.	Separate the user interaction model
Make it easy to add or remove features	Use vendor-independent interfaces.	Separate time-critical components

Several factors may influence the same objectives, so strategies should be developed that balance these factors and benefit from their synergy. Each strategy needs to be consistent with the characteristics of the influencing factors, their desired/required changeability, and their interactions with other factors. Strategies should address one or more of the following goals in order to ensure the buildability and changeability of the architecture design: reduce or localize the factor's influence, reduce the impact of the factor's changeability on the design and other factors, reduce or localize required areas of expertise or skills, reduce overall time and effort.

After reviewing the analysis for the example system, we identified objectives such as: meeting an aggressive schedule, upgrading domain-specific hardware, easy addition or removal of features, and high performance. Table 4 gives an overview of some strategies to address the "Easy Addition or Removal of Features" objective.

Table 4: Objective - Easy Addition or Removal of Features

Influencing Factors	*Applicable Strategies*
Time to market is short.	Use a flexible pipeline model.
Delivery of features is negotiable.	Separate components along dimensions of concern.
New features can be added every three years.	Encapsulate features into separate components.
User interaction model must be adapted to new standards.	

Designing the architecture to make it easy to add or remove features was motivated to help meet the aggressive schedule by trading off non-essential functionality with time. However, designing a system for easy addition or removal of features is a non-trivial problem. The solution uses the principle of separation of concerns to address the problem. We developed a flexible pipeline model for building acquisition and processing applications. Acquisition and processing components can be used as stages in the pipeline. This will allow us to quickly introduce new acquisition types by constructing pipelines using old and new components. We followed the strategy of separating components along dimensions of concern to build in flexibility to accommodate change in the architecture. We separated or decomposed modules along important dimensions of concerns, including processing, communication, control, data, and user interface aspects of the software design. This gives us the possibility of using the modules in other applications or in contexts we cannot foresee at this time. Application of this strategy will also allow us to allocate and trace the requirements to the design elements. When the requirements change, it will be easier to re-

use the existing framework and plug in new components to get a new solution more quickly. To isolate the effects of change to product features, we encapsulated related product features into separate components (e.g., movement of the probe, sensor processing, connectivity to the network).

3 CONCLUSIONS

To take into account the factors that influence the architecture, the architect considers and balances a number of strategies to come up with the appropriate architectural decisions. In addition to implementation strategies that ensure that developers can use the architecture to guide their implementation of the system, there are changeability strategies that ensure that the architecture will be able to provide a stable base that can accommodate or localize the effects of changes in the organization, technology, and product factors.

We have found from our experience that successful architects prepare for likely changes. They do this by noting the flexibility of the influencing factors and how likely they are to change. They note how factors interact and what impact they have in order to prioritize and evaluate what strategies are most cost effective in preparing for change. We have observed this kind of analysis informally in a number of systems. We are in the process of describing four industrial software systems using this global analysis approach (Hofmeister et al.).

4 REFERENCES

Bass, L., Clements, P., and Kazman, R. (1998) *Software Architecture in Practice*. Addison-Wesley, Reading, Massachusetts.

Clements, P.C. (1995) Understanding Architectural Influences and Decisions in Large System Projects, in *Proceedings of the First International Workshop on Architectures for Software Systems*, Seattle, WA.

Hofmeister, C., Nord, R., Soni, D. (to appear) *Applied Software Architecture*. Addison-Wesley, Reading, Massachusetts.

Parnas, D. (1972) On the Criteria for Decomposing Systems into Modules. *Communications of the ACM* 15, 12, 1053-1058.

Soni, D., Nord, R.L., and Hofmeister, C. (1995) Software Architecture in Industrial Applications, in *Proceedings of the 17th International Conference on Software Engineering*, Seattle, WA.