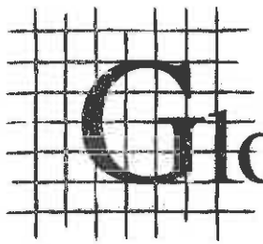
processing filters may be added. The product needs to remain compatible with new or evolving standards for file formats and communication of image information.

In addition, the technology affecting the software components is likely to change over the lifetime of the system. Even if the requirements for the functionality of an image processing filter don't change, its implementation might change, for example, to improve performance. IS2000 may have to be improved to handle upgrades to commercial components that are part of the product, and the target software environment is likely to change as upgrades are introduced.

In the next five chapters, we present how the design of IS2000's software architecture meets these challenges. Some of the design constraints are ones that every software designer faces and some of them are unique to the IS2000 functionality and marketplace.



Chapter 3
Global Analysis

The purpose of global analysis is to analyze the factors that influence the architecture and to develop strategies for accommodating these factors in the architecture design. Global analysis starts before the conceptual, module, execution, and code architecture views are defined, and it continues throughout the architecture design.

Global analysis begins with identifying factors that affect the architecture design. These influencing factors fall into three categories:

1. Organizational factors
2. Technological factors
3. Product factors

Organizational factors constrain the design choices. Some of them, such as development schedule and budget, apply only to the product you are currently designing. Other organizational factors, such as organizational attitudes and the software process, carry over to every product developed by an organization. If you ignore these factors, the architecture may not be able to be implemented.

The technological factors have a more obvious influence on the architecture. Your design choices are limited by the kinds of hardware and software, architecture technology, and standards that are currently available. But the state of technology changes over time, and products must adapt, so your architecture should be designed with flexibility in mind.

The product factors are the primary influence on the architecture. This category covers the functional features of your product, and qualities like performance, dependability, and cost. These factors may be different in future versions of the product, so your architecture should be designed to support the changes you can reasonably expect.

We call this activity *global* analysis for three reasons:

1. Many of the factors affect the entire system; they can't necessarily be localized in one component of the architecture. The strategies should address the global concerns, but provide guidance for implementing them locally.
2. Global analysis occurs throughout the architecture design rather than at one point in time. As you design each of the architecture views, you focus on developing strategies that help you design that view. New factors, issues, or strategies can arise at any time during the design, as the details of the architecture are pinned down.
3. During global analysis you consider the factors as a group instead of independently. These factors are not all independent; some of them may even contradict each other. You must sort out the conflicts and resolve them, either by prioritizing the factors or by negotiating changes to factors when you can.

Global analysis is meant to complement the risk analysis and/or requirements analysis that you perform—not replace it. It's possible that after completing the requirements and risk analyses you will already have a thorough analysis of the factors that affect the architecture. Then the focus of global analysis is to develop strategies for the architecture design.

The primary purpose of global analysis is to give you a systematic way of identifying, accommodating, and describing the factors that affect the architecture. The three types of factors are analyzed separately, with the results recorded in separate *factor tables*. You then develop strategies for accommodating the factors, and you describe them on *issue cards*.

In this chapter we describe this analysis and apply it to IS2000. We analyze the organizational factors first, then the technological and product factors. This is probably the reverse of the order you will address them, but we wanted to introduce the less obvious factors first, ending with the more familiar product factors.

3.1 Overview of Global Analysis Activities

The architecture design of a large software system is influenced by a variety of factors. Many of these have an impact over the entire system, and some directly contradict other factors. Some factors, such as requirements for fault tolerance, can affect the design in a fundamental manner. To avoid expensive rework, these factors must be addressed from the beginning of design. There is a tension between factors that constrain the design and factors that make it more flexible. These factors have to be balanced so that the system is able to be implemented and adapted to future needs.

During global analysis, you uncover the most influential of these factors, then develop strategies for designing the architecture so that it accommodates these factors and reflects global and future concerns. You should record the results of the analysis process as a part of the architecture so that individual developers can use these results while making decisions to address specific design problems. Without strategies that reflect global and future

concerns, developers may choose a local solution that does not support anticipated changes. As the architecture design proceeds, the architect should monitor the efficacy and relevance of these strategies, and make changes when necessary.

To arrive at these strategies, there are two basic activities—analyze factors and develop strategies—each with three steps (Figure 3.1). Global analysis is the first task in each of the four architecture views, and its results are used in the central design tasks of that view. As you see in Figure 3.1, during the central design tasks, new issues or strategies may arise that cause you to return to the global analysis and to revise or expand your factor tables or issue cards.

3.1.1 Analyze Factors

The analyze factors activity takes as input the organizational, technological, and product factors. If these factors are not yet explicit, then part of the first step is to make them explicit.

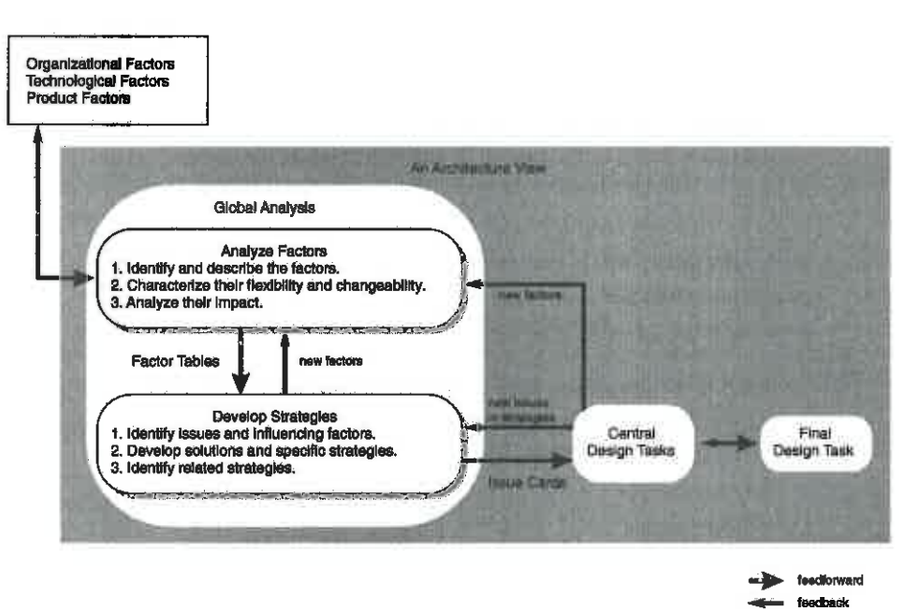


Figure 3.1. Global analysis activities

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 influence, those that could change during development, those that are difficult to satisfy, and those with which you have little experience.

To determine whether a factor has significant global influence, you 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 is the factor important?
- Does the factor require any new expertise or skills?

Step 2: Characterize the Flexibility or the Changeability of the Factors

The flexibility of a factor describes what is negotiable about the factor. The negotiating could be with any of the stakeholders: your managers, marketing personnel, the customers, the users, the system architect, or the developers. Use this information when factors conflict or for some other reason become impossible to fulfill.

To characterize the flexibility of a factor, you should ask

- Is it possible to influence or change the factor so that it makes your task of architecture development easier?
- In what way can you influence it?
- To what extent can you influence it?

The changeability of a factor describes what you expect could change about the factor, both in the near and more distant future. The changeability could be due to a change in the factor itself, or because of changes in the way you expect to use it in the future.

To characterize the changeability of a factor, you should ask

- In what way could the factor change?
- How likely will it change during or after development?
- How often will it change?
- Will the factor be affected by changes in other factors?

Step 3: Analyze the Impact of the Factors

Here you analyze the impact of the factors on the architecture. If the factor was to change, which of the following would be affected and how:

- Other factors
- Components
- Modes of operation of the system
- Other design decisions

You should record the impact of factors and their changeability in terms of impact on particular components, but this won't be possible if the components have not yet been defined or identified. Some experienced architects, even at this early stage, have a notion of the high-level components of the system. But even if you're making guesses during the early stages, it is important to capture your assumptions to get feedback as soon as possible.

As the design progresses, the impact of factors can be determined more precisely, and you should revise the documentation accordingly. We recommend that you use the format presented in Table 3.1, which we call a *factor table*, for recording the results of analyzing the factors. Keep three tables, one for each type of factor (organizational, technological, and product). We use a label for each factor that identifies its factor category (O1 or O2 in Table 3.1), and a unique sequencing number (O1.1, O1.2, and so on). The labels for organizational factors begin with the letter O, technological factors begin with T, and product factors begin with P.

Organizational Factor	Flexibility and Changeability	Impact
O1: <Factor category>		
O1.1: <Factor name>		
<Description of factor>	<What aspects of the factor are flexible or changeable?>	<Components affected by the factor or changes to it>
O1.2: <Factor name>		
<Description of factor>	<What aspects of the factor are flexible or changeable?>	<Components affected by the factor or changes to it>
O2: <Factor category>		
O2.1: <Factor name>		
<Description of factor>	<What aspects of the factor are flexible or changeable?>	<Components affected by the factor or changes to it>

Table 3.1. Sample Format of Organizational Factor Table

3.1.2 Develop Strategies

The second global analysis activity is to develop strategies for the architecture design or for implementation. This activity also has three steps.

Step 1: Identify Issues and Influencing Factors

Using the factor tables, you first identify a handful of important issues that are influenced by the factors and their changeability.

- An issue may arise from limitations or constraints imposed by factors. For example, an aggressive development schedule may make it impossible to satisfy all of the product requirements.
- An issue may result from the need to reduce the impact of changeability of factors. For example, it is often important to design the architecture to reduce the cost of porting the product to another operating system.
- An issue may develop because of the difficulty in satisfying product factors. For example, a high throughput requirement may overload the CPU.
- An issue may arise from the need to have a common solution to global requirements such as error handling and recovery.

Usually there are several factors that affect an issue, and when these factors conflict, or when fulfilling all of them is too expensive, then you must negotiate a change in the factors.

Step 2: Develop Solutions and Specific Strategies

For each issue, develop strategies that address the issue and ensure the implementation and changeability of the architecture design. Design each strategy to be consistent with the influencing factors, their desired/required changeability, and their interactions with other factors. Your strategies should address one or more of the following goals:

- 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.

Step 3: Identify Related Strategies

When a strategy belongs with more than one issue, don't repeat the strategy: Describe it in just one place, and reference it as a related strategy in the other issues where it applies.

Use a standard format to describe each issue, its influencing factors and their impact, a general discussion of its solution, and the strategies that address it. Here is the format we recommend for describing an architecture design issue: We call this an *issue card*.

<Name of the Architecture Design Issue>

<Description of the issue>

Influencing Factors

<List of the factors that affect this design issue and how>

Solution

<Discussion of a general solution to the design issue, followed by a list of the associated strategies>

Strategy: *<Name of the strategy>*

<Explanation of the strategy>

Related Strategies

<References to related strategies and a discussion of how they are related to this design issue>

Make sure you use meaningful names to capture the essence of strategies and to make it easy to communicate. The importance of communication can't be overemphasized when it comes to strategies that will be applied by other developers.

You may want to summarize your strategies with a table that cross-indexes issues, influencing factors, and strategies (look ahead to Table 3.5 for an example). This overview acts as a guide for developers to find and to apply the appropriate strategy.

3.2 Analyze Organizational Factors

Next we present a detailed example of applying the global analysis to IS2000, starting with the analysis of the organizational factors. Certain factors that arise from the business organization itself affect the design of the architecture. We call these factors *organizational factors*. They include aspects of the project such as the schedule and budget, and the skills and interests of the people involved.

In Figure 3.2, we identified five categories of organizational factors: management, staff skills, process and development environment, development schedule, and development budget. Within these five categories we listed examples of typical organizational factors.

O1: Management
Build versus buy
Schedule versus functionality
Environment
Business goals

O2: Staff skills, interests, strengths, weaknesses
Application domain
Software design
Specialized implementation techniques
Specialized analysis techniques

O3: Process and development environment
Development platform
Development process and tools
Configuration management process and tools
Production process and tools
Testing process and tools
Release process and tools

O4: Development schedule
Time-to-market
Delivery of features
Release schedule

O5: Development budget
Head count
Cost of development tools

Figure 3.2. Typical organizational factors

The organizational factors don't describe the product; instead, they capture aspects of the organization that could affect the architecture. Although the factors in Figure 3.2 are the concerns of a project manager, they also concern an architect because of their potential impact on the architecture. They constrain the design choices while the product is being designed and built, and if they are ignored, the architecture may not be able to be implemented. For example, if the schedule is aggressive, there may not be enough time to train developers in a new technology, which rules it out for the architecture design.

For most of these factors, your primary concern is to determine their flexibility, to help you negotiate the constraints if necessary. If a factor is likely to change during development, due to a change in management or market conditions for example, you should also analyze its changeability.

Step 1: Identify and Describe the Factors

The first step in analyzing organizational factors is to identify the factors that may affect the architecture. Table 3.2 describes some of the organizational factors for IS2000. The factors in the management section reflect the organization's historic preference for developing software in-house rather than acquiring it outside, and the high importance of meeting the schedule, even at the expense of not providing all the functionality. For other systems, organizational attitudes may not be relevant, or they may neutral enough to have no effect on the architecture.

For the staff skills section, we identified the skills and the experience developers should have for this project, and compared them with the actual skills. Early in the architecture design, you won't always know which specialized skills are needed. In this case you have to make a prediction, and revise these factors as the architecture design progresses.

For IS2000, the schedule factors have a significant impact. The product has a two-year time-to-market, and the features to be delivered are prioritized. It is clear to all the stakeholders that this is a very aggressive schedule, so they agreed early to a prioritization of features.

Step 2: Determine the Flexibility and the Changeability of the Factors

The next step in analyzing organizational factors is to characterize the flexibility and changeability of each factor, keeping in mind the priorities set by management and the organizational attitudes.

For IS2000, the results of this step are shown in the middle column of Table 3.2. The company considers time-to-market (factor O4.1) more important than development cost (factor O5.1) or delivering all planned features (factor O4.2). The organization's attitudes (factors O1.1 and O1.2) are relatively neutral, and do not preclude any architectural design choices.

Note that the in-house personnel assignments were made before the architecture design began. These are fixed, but there is some flexibility in either hiring a few permanent employees or hiring contractors for the duration of the project.

Step 3: Analyze the Impact of the Factors

The third step is to analyze the impact of the organizational factors and their flexibility or changeability. The flexibility of management preferences can be used, for example, to postpone implementation of product features to meet the schedule. The skill deficiency of the staff in use of multiple threads and processes can have a severe impact on the ability of the system to meet its performance requirements. The schedule and flexible delivery of features have a large and pervasive impact on almost all of the architecture design.

Organizational Factor	Flexibility and Changeability	Impact
O1: Management		
O1.1: Build versus buy		
There is a mild preference to build.	The organization will consider buying if it is justified.	There is a moderate impact on meeting the schedule.
O1.2: Schedule versus functionality		
There is a preference for meeting the schedule over some features.	This is negotiable for several features.	There is a moderate impact on meeting the schedule and the first product release.
O2: Staff skills		
O2.1: Experience in structured design		
All in-house developers have these skills.	Flexibility is not applicable.	There is a small impact.
O2.2: Experience in object-oriented analysis and design		
Half of the in-house developers have these skills.	It is feasible to hold training.	There is a moderate impact on achieving good design.
O2.3: Experience in multithreading		
One in-house developer has this skill.	The subject area is too complex to rely on training alone.	There is a moderate impact on meeting performance.
O2.4: Experience in building multiprocess systems		
Two in-house developers have this skill.	Training supplemented with software abstraction may alleviate lack of skills.	There is a large impact on meeting performance.

Table 3.2. Organizational Factors for IS2000

Organizational Factor	Flexibility and Changeability	Impact
O4: Development schedule		
O4.1: Time-to-market		
Time-to-market is two years.	There is no flexibility.	There is a large impact on design choices in all areas.
O4.2: Delivery of features		
The features are prioritized.	The features are negotiable.	There is a moderate impact on meeting the schedule.
O5: Development budget		
O5.1: Head count		
There are 12 developers.	The organization can hire one or two permanent developers or a large number of contractors.	There is a moderate impact on meeting the schedule.

Table 3.2. Organizational Factors for IS2000 *(continued)*

3.3 Begin Developing Strategies

After analyzing some of the factors you can begin identifying key issues in the architecture design and determine whether any solution strategies are emerging. Later in the global analysis, as you consider the technological factors and product requirements, you refine or revise these strategies. In practice, as you design each architectural view, you may need to identify additional strategies to handle design problems that become apparent only after some of the design has been completed.

Step 1: Identify Issues and Influencing Factors

After reviewing the analysis summarized in Table 3.2, we identified two issues: Aggressive Schedule and Skill Deficiencies. Obviously we can't know for certain the aggressive-

ness of the schedule or what skills are lacking without knowing certain key product factors, without having decided certain aspects of the architecture design, and without having at least a rough estimate of the resources needed to implement the product. At this point in the analysis we rely on past experience with similar products to identify the key issues for this product. It is possible that these issues will be addessed as we proceed further with the analysis and architecture design, but more likely we will refine the issue and modify the strategies for resolving it.

For the Aggressive Schedule issue, most of the influencing factors are organizational factors, but factor P7.2, COTS budget, is a product factor. This factor comes up later in our analysis, but we have included it here so we don't have to repeat the issue card again later.

Step 2: Develop Solutions and Specific Strategies

We propose three strategies to solve the Aggressive Schedule issue. The first two strategies mandate that we use existing software components whenever possible, either in-house or commercial software. The second strategy relies on the fact that features are prioritized and there is some flexibility in their delivery dates. If we design the architecture so that it is easy to add or to remove features, then we can adjust the feature set to meet the delivery dates.

For the Skill Deficiencies issue, past experience indicates that a system like this must be a multiprocess system. However, the development team overall is weak in the necessary skills. Thus the two strategies we propose are (1) to avoid using multiple threads and (2) to encapsulate multiprocess support. Later analysis may show these strategies to be infeasible or may support them.

Experienced architects begin to consider strategies such as these very early in the design process. They reconsider them as the architecture design progresses, so it is important to capture not only the initial strategies, but also the factors that prompt them. Having a record of the influencing factors gives you more information about the consequences of revising a strategy.

Step 3: Identify Related Strategies

At this point, for these two issues, there are no related strategies. Later we present other issues that are related to them.

Aggressive Schedule

The development schedule is aggressive. Given the estimated effort and available resources, it may not be possible to develop all the software in the required time.

Influencing Factors

O4.1: Time-to-market is short and is not negotiable. A rough estimate of effort required to redesign and reimplement all of the software suggests that it will take longer than two years.

O5.1: Head count cannot be increased substantially.

O1.1: Building is mildly preferred over buying.

O4.2: Delivery of features is negotiable. Low-priority features can be added to later releases.

P7.2: Budget for commercial off-the-shelf (COTS) components is flexible. Both the price and the licensing fees of COTS components must be considered.

Solution

Redesigning and reimplementing all of the software will take longer than two years. Three possible strategies are to reuse software, buy COTS, and to release low-priority features at a later stage.

Strategy: *Reuse existing in-house, domain-specific components.*

Several of the in-house domain-specific components are candidates for reuse. However, reuse of some existing components may need substantial redesign and reimplementation. Evaluate each of these components to determine whether it is advantageous to reuse it and whether it will save time and effort.

Strategy: *Buy rather than build.*

Buying COTS software has the potential of saving time and effort. However, the price and licensing fees for some COTS products may be too high. Learning to use new COTS software may increase time and effort. Purchase or license COTS software when it is advantageous and when it will reduce development time substantially.

Strategy: *Make it easy to add or remove features.*

One way to reduce the develop time is to reduce the functionality by delaying delivery of some of the features to a later release. If it is easy to add or to remove functional features without substantial reimplementation, then it is feasible to adjust the functionality to meet the delivery schedule.

Skill Deficiencies

We know from experience with the previous product that it will be a challenge to meet this product's performance requirements, which are considerably tighter than for the prior system. Common techniques to achieve higher performance are to use multiple threads and multiple processes. However, the development team is deficient in the necessary skills.

Influencing Factors

O2.3: There is only one developer who has experience with the use of multiple threads.

O2.4: There are only two developers who have experience with the use of multiple processes.

Solution

Two possible strategies are to avoid the use of multithreading and to encapsulate multiprocess facilities.

Strategy: ***Avoid the use of multiple threads.***

Multithreading can be quite complex, difficult to use, and error prone. A training course alone would not be sufficient for a developer to achieve proficiency in multithreading. Rather than hiring someone with multithreading experience, avoid multithreading whenever possible and use it only when absolutely necessary.

Strategy: ***Encapsulate multiprocess support facilities.***

Because we have very few people with multiprocessing skills, our strategy is to maximize the available skills by encapsulating the multiprocess support facilities in a layer, and to provide a simpler interface that's tailored to the specific project needs.

3.4 Analyze Technological Factors

We now continue the global analysis with the technological factors. These factors arise from the external technology solutions that are embedded or embodied in the product. They are primarily hardware and software technologies and standards. Typical categories of technological factors are general-purpose and domain-specific hardware, software technologies such as the operating system and user interface, architecture technologies such as patterns and frameworks, and standards such as image data formats (Figure 3.3). For each of these categories, consider whether there are general-purpose and/or domain-specific

solutions appropriate for your product. Also consider whether the available solutions are commercially available or belong to the company, whether for a product family or a specific product.

T1: General-purpose hardware
Processors
Network
Memory
Disk

T2: Domain-specific hardware
Probe hardware
Probe network

T3: Software technology
Operating system
User interface
Software components
Implementation language
Design patterns
Frameworks

T4: Architecture technology
Architecture styles
Architecture patterns and frameworks
Domain-specific or reference architectures
Architecture description languages
Product-line technologies

T5: Standards
Operating system interface
Database
Data formats
Communication
Algorithms and techniques
Coding conventions

Figure 3.3. Typical categories of technological factors

Although, like the organizational factors, these factors do not describe the product itself, its design and implementation directly depends on them. These factors are likely to change, which in turn forces the design and implementation of the software system to change accordingly. Your architecture should be designed to minimize the difficulty of adapting to expected changes in technological factors.

In analyzing the technological factors, you follow the same three steps as for the organizational factors, and produce a factor table that summarizes the technological factors. These steps should be clear by now, so for IS2000, let's go directly to the technological factor table (Table 3.3).

Technological Factor	Flexibility and Changeability	Impact
T1: General-purpose hardware		
T1.1: Processor type		
A standard processor has been selected.	Increases in processor speed are frequent. As technology improves, the processor type could change every four years.	The change in processor type is expected to be transparent, provided the operating system does not change.
T1.2: Number of processors		
Only one processor has been deemed to be sufficient initially.	If any additional functionality or performance is required, and available processor speeds do not increase, an additional processor may be required.	The change can be transparent if the operating system (OS) supports multiprocessing. If not, components at processor boundaries will be affected drastically.

Table 3.3. Technological Factors for IS2000

Technological Factor	Flexibility and Changeability	Impact
T1.3: Memory		
A total of 64MB of memory has been selected due to constraints in budget.	If memory prices drop, then this size could be increased. No significant decrease is expected during development.	More complex signal processing may become possible when memory size can be increased.
T2: Domain-specific hardware		
T2.1: Probe hardware		
This is the hardware to detect and process signals.	The probe hardware may be upgraded every three years as technology improves.	The impact will be large on components involved in acquisition and image processing.
T2.2: Probe network		
This is the network connecting components of the probe hardware and general-purpose hardware.	The probe network is expected to change every four years as technology improves.	The impact will be large on components involved in acquisition and image processing.
T3: Software technology		
T3.1: Operating system		
The OS on the general-purpose processor is a real-time OS. A nonreal-time OS will be used on additional CPUs if they are deployed.	OS features change every two years. The OS itself may change every four years.	Changes are transparent provided they conform to the current standard for OS interface. Otherwise, the impact will be large.

Table 3.3. Technological Factors for IS2000 *(continued)*

Technological Factor	Flexibility and Changeability	Impact
T3.2: OS processes		
Memory overhead and overhead for creation/ destruction of a process are important considerations. Overhead for creation/destruction of processes is small on the real-time OS.	Overheads will be reduced with improvements in OS implementations. They may increase for a new OS, however.	The impact is large on allocation of modules to processes and threads in the execution view.
T5: Standards		
T5.1: Standard for OS interface		
POSIX has been selected as the OS interface standard.	This standard is stable.	There is large impact on all components.
T5.2: Standards for sensor data format		
The standards affect the format of the data available from the probe hardware.	There is no stable standard for this format.	There is a large impact on components involved in acquisition, image processing, and postprocessing.
T5.3: Standards for image data formats		
The format of image data is standardized in this domain.	The standard is somewhat stable, but is likely to change every three years.	There is a large impact on image-processing and postprocessing components.

Table 3.3. Technological Factors for IS2000 *(continued)*

Technological Factor	Flexibility and Changeability	Impact
T5.4: Domain-specific algorithms and techniques		
These are the algorithms for signal processing, image construction, and image processing.	The field is quite mature. Improvements will be minor and infrequent.	There is a minor impact on components involved in acquisition, image processing, and postprocessing.

Table 3.3. Technological Factors for IS2000 *(continued)*

For the general-purpose hardware category we have three factors: T1.1 and T1.2 record our initial choice of a single standard processor, which we expect to provide sufficient computing power, and T1.3 records our 64MB memory size. The general-purpose hardware platform is one of the major technological factors that is likely to change. As technology improves, the goal is to take advantage of faster processors or even move to a multiprocessor architecture. At this point a single CPU is planned, but we may add a second if one CPU cannot cope with the expected system load. To keep costs down, though, the memory limit is pretty firm.

Next let's consider the implications of a change in the processor type. Because the operating system provides a layer over the hardware, such a change will likely be transparent, provided the hardware change does not require a change to a different operating system. Adding one or more processors would mean adding functionality to support multiprocessing. This change would have a large impact on all components at the processor boundaries.

For domain-specific hardware, we have the probe hardware (factor T2.1) and probe network (factor T2.2). Here we need to build in flexibility to introduce new models of the probe and probe networks, and to specify different hardware configurations (for example, for low-end to high-end products). Achieving this kind of flexibility will be a challenge, because changes to either the probe hardware or the network will affect all components involved in the acquisition or image processing.

In the software technology category, there are factors for the operating system (factor T3.1) and operating system processes (factor T3.2). The operating system is likely to change, for example if the requirements are updated to include support for new hardware platforms. We expect this change to be transparent, provided the operating system still conforms to our selected interface standard.

For the operating system processes, we don't expect to run up against the limit for the number of operating system processes, but each operating system process uses resources that are limited. If, during development, the system meets these limits, we may have to set limits on the use of operating system processes.

The final category, standards, lists the standards related to the operating system interface (factor T5.1), sensor data format (factor T5.2), image formats (factor T5.3), and domain-specific algorithms (factor T5.4). The system is connected to a network that transports the domain-specific data. Here we want to take advantage of vendor solutions, using standards when available so that we can upgrade to new versions. We also need to use the interface standard available for the sensor data format, although it is much less stable.

A change to the standards for sensor and image data formats would affect both acquisition and image-processing components as well as the postprocessing components (factors T5.2 and T5.3). If, however, the standard for the operating system interface (factor T5.1) were to change, it would have a large impact on all components of the system.

We want to incorporate standard domain-specific techniques (for example, edge detection in image processing), and to be prepared to upgrade them as appropriate. Because the application domain of the product is mature, the algorithms and techniques for this product are the same ones that have been used in prior products. They are unlikely to undergo any major changes during the life of the product. This is fortunate because such a change could affect components involved in acquisition, image processing, and postprocessing, although the impact would be localized to the particular algorithm or technique (factor T5.4).

3.5 Continue Developing Strategies

The next step is to review the analysis thus far to identify key issues that arise from the technological factors and their changeability, and to develop strategies to address them. Some of the strategies can introduce new technological factors, which causes us to revisit some of the previous steps to assess the impact of these newly introduced factors. Later in this chapter, when we consider the product requirements, we may refine or revise these strategies.

In reviewing the analysis summarized in Table 3.3, we identify the following two issues:

1. **Changes in General-Purpose and Domain-Specific Hardware**—Changes in both the general-purpose and the domain-specific hardware are anticipated on a regular basis. The problem is to adapt to these changes quickly.
2. **Changes in Software Technology**—Changes in software technology will affect several important components. The problem is to reduce the effort in adapting the affected components.

Changes in General-Purpose and Domain-Specific Hardware

Changes in both the general-purpose and the domain-specific hardware are anticipated on a regular basis. The challenge is to reduce the effort and time involved in adapting the product to the new hardware.

Influencing Factors

T1.1: The processor speed is likely to change very frequently, even during development. As technology improves, the goal is to take advantage of faster processors or even move to a multiprocessor architecture.

T2.1: The probe hardware is expected to change every three years. Changes in the probe hardware can change performance requirements as well as sensor data formats. This, in turn, affects the components involved in acquisition and image processing.

T2.2: The probe network is expected to change every four years. This may change the throughput of signal data, and therefore the acquisition components. The higher data rate may also affect the memory requirements of image-processing components.

Solution

Separate the software that directly interacts with the hardware.

The following strategies should be applied first in the conceptual view to introduce components that encapsulate the hardware and to separate them from the application components. They should then be applied in the module view to encapsulate the software related to conceptual components in corresponding modules. The strategies are also useful in separating software related to hardware components from software related to application components.

Strategy: *Encapsulate domain-specific hardware.*

The system interacts with the domain-specific hardware of the probe. The system should be flexible in allowing for new models of the probe to be introduced and new hardware configurations to be specified. Use an abstraction for the probe to minimize the effects of any changes to the probe hardware.

Strategy: *Encapsulate general-purpose hardware.*

Encapsulate the system hardware to allow changes to be made to the hardware with little or no impact on the applications. Conversely, this strategy will support the introduction of new application features without requiring modifications to the software that manages the hardware.

Changes in Software Technology

Off-the-shelf components such as the operating system have a large impact on a significant number of system components. The challenge is to reduce the effort and time necessary to adapt the system when these changes arise.

Influencing Factors

T3.1: Operating system features change every two years. The operating system itself may change every four years. The impact of these changes is large unless they conform to the selected operating system interface standard.

T3.2: Changes in operating system process models have a large impact on the allocation of modules to processes and threads in the execution view.

T5: Standards exist for many of the external components. In some cases such standards are unstable, whereas in others they do not exist at all.

Solution

Use standards when possible and develop internal, product-specific interfaces to commercial off-the-shelf components.

Strategy: *Use standards.*

Use standards when possible to reduce the impact of changes on software technology. Use a standard operating system interface such as POSIX to facilitate porting to another operating system in the future.

Strategy: *Develop product-specific interfaces to external components.*

When standards are unstable or absent, create internal standards. Develop product-specific interfaces to reduce dependencies on external components and unstable standards. A good candidate for the application of this strategy is sensor communication.

Related Strategies

Related and synergistic strategies are *Buy rather than build* and *Reuse existing in-house, domain-specific components* (issue, Aggressive Schedule).

3.6 Analyze Product Factors

The remaining type of factors to be analyzed are the product factors. They describe the product's requirements for functionality, the features seen by the user, and qualities such as performance. Figure 3.4 shows typical categories of requirements: functional features; user interface; performance; dependability; failure detection, reporting, recovery; service; and product cost.

P1: Functional features
Functional features

P2: User interface
User interaction model
User interface features

P3: Performance
Acquisition performance
Sensor data rate
Start-up and shutdown times
Recovery time

P4: Dependability
Availability
Reliability
Safety

P5: Failure detection, reporting, recovery
Error classification
Error logging
Diagnostics
Recovery

P6: Service
Service features
Software installation and upgrade
Maintenance of domain-specific hardware
Software testing
Maintenance of software

P7: Product Cost
Hardware budget
Software licensing budget

Figure 3.4. Typical categories of product factors

The product factors have a larger impact on the architecture than either the organizational factors or the technological factors. As the market changes, the product factors may also change, sometimes drastically. The product factors can also change during development. The architecture should be designed so that it can satisfy the requirements and easily accommodate anticipated changes.

Although we emphasized flexibility with the organizational factors and changeability with the technological factors, here we need to characterize both the flexibility and the changeability of the product factors. Examine the flexibility of product factors to ensure the implementation of your design, and examine the changeability of requirements to ensure system evolution.

In analyzing the product factors, we again follow the process outlined in Section 3.1. The results of applying this to IS2000 are shown in Table 3.4.

This system converts raw data into images and provides a range of image-processing operations on two- and three-dimensional images. The user sets up parameters for an acquisition procedure, then the system runs the acquisition interactively. The system normally acquires data without interruption, but the user can interrupt and exert some control over the acquisition. The user can also select from acquisition and image-processing algorithms to balance the acquisition speed and the image quality.

Product Factor	Flexibility and Changeability	Impact
P1: Functional features		
P1.1: Acquisition procedures		
Acquire raw signal data and convert it into two- and three-dimensional images. The system has a number of standard acquisition procedures.	New acquisition procedures may be added every three years.	This feature affects acquisition performance, image processing, and the user interface.
P1.2: Image processing		
A range of two- and three-dimensional image-processing algorithms are supported.	New image-processing algorithms can be added on a regular basis.	This feature affects the user interface and acquisition performance.

Table 3.4. Product Factors for IS2000

Product Factor	Flexibility and Changeability	Impact
P1.3: Image types		
A range of image types are supported.	New image types can be added with new processing algorithms.	This feature affects persistence of storage, acquisition and image-processing components, and communication over a network.
P2: User interface		
P2.1: User interaction model		
The user can control image acquisition interactively.	This feature must adapt to new paradigms and to new domain standards every three years.	This feature affects the user interface component, and may affect the acquisition and storage components.
P2.2: User-level acquisition control		
The user can set up parameters for an acquisition procedure, select acquisition algorithms, and start, pause, and stop the acquisition.	Requirements for user-level acquisition control are stable.	This feature affects the acquisition and storage components.
P3: Performance		
P3.1: Maximum signal data rate		
This is the rate at which the probe can acquire data.	The maximum data rate changes with changes in the probe hardware.	This factor affects acquisition performance.

Table 3.4. Product Factors for IS2000 *(continued)*

Product Factor	Flexibility and Changeability	Impact
P3.2: Acquisition performance		
Acquisition performance is measured by the size and number of images, and acquisition response time is measured in terms of end-to-end deadlines.	The acquisition performance requirements are slightly flexible. Their effect on performance requirements of individual components is likely to change during development when the system is tuned or whenever the product is modified.	A large impact on all components involved in acquisition and image processing, storage, and display can be expected.
P7: Product cost		
P7.1: General-purpose hardware budget		
The budget for the general-purpose hardware is limited and the part allocated to memory restricts the maximum memory size to 64MB.	There is no flexibility in the budget.	The budget has a moderate impact on the components for acquisition and image processing.
P7.2: Commercial off-the-shelf (COTS) budget		
The maximum limit for licensing COTS software is $2,000.	There is some flexibility if time-to-market can be reduced.	There is a moderate impact on meeting the schedule.

Table 3.4. Product Factors for IS2000 *(continued)*

Advances in hardware, software, and the application domain will likely drive changes in requirements as technology enables the user to do new things. We expect to add or to enhance acquisition procedures and algorithms every few years. New image-processing algorithms are expected to be added on a regular basis. Adding these new features may affect the system's performance.

The user interaction model is likely to change as users want to perform more sophisticated tasks and, at the same time, have the system take on more of the administrative functions, thereby making the system easier to use. Changes to the user interaction model will affect the system interface.

For IS2000, the performance factors describe the maximum rate at which the signal data can be acquired by the probe hardware, and they describe acquisition performance. The maximum signal data rate is expected to change every few years as faster versions of the probe hardware become available.

There are several different measures of the acquisition performance of a system. The system must keep up with the raw events coming into the system, process them, and then make them available for viewing in a timely manner. Responsibility for meeting this requirement is completely distributed (there are no central aspects), although the requirement doesn't necessarily affect the entire system. To meet this requirement, we need to acquire people who have expertise in data acquisition and image processing, communications, control/coordination, and knowledge of the performance properties of the hardware and software platforms.

The requirements for acquisition performance will change when the maximum data rate changes or when new acquisition procedures and image-processing algorithms are offered. Responsibility for acquisition is shared among a wide cross-section of components, each of which is assigned a performance budget. The overall acquisition performance is dependent on such performance budgets, the performance of the software platform (for example, performance of communication mechanisms), and decisions related to process and processor boundaries. These parameters are likely to change as the system is tuned during development and when performance bottlenecks occur, when the system can be run at full capacity, and after initial performance experiments. The acquisition performance requirements are slightly flexible and can be negotiated if changes are small.

The impact of the performance requirements is expected to be large. We need to reduce the scope of this impact to the minimal number of components, because satisfying real-time requirements requires more overhead (for example, special hardware, a real-time software platform, stringent guidelines/constraints on components, and rigorous analysis) and introduces more complexity and higher risk to the project.

3.7 Continue Developing Strategies

Again let's try to identify new issues using the additional information about the product factors and the strategies we've developed so far. Some of the strategies could introduce or modify product factors. In this case, we need to revisit some of the previous steps to assess the impact of the new factor.

Because of the changeability of the functional features and the user interaction model, we identified the issue of Easy Addition and Removal of Features. This also arises from the strategy *Make it easy to add and remove features* (from the Aggressive Schedule issue), which is a very general strategy. By raising it to the level of an issue, we now add more specific strategies for achieving it.

Easy Addition and Removal of Features

Making it easy to add or remove features will help meet the aggressive schedule by trading off function with time. However, designing a system for easy addition and removal of features is a nontrivial problem.

Influencing Factors

O4.1: Time-to-market is short.

O4.2: Delivery of features is negotiable.

P1: New varieties of features may be added every three years.

P2.1: The user interaction model must be adapted to new paradigms and standards.

Solution

Use the principle of separation of concerns to develop specific strategies to address particular problems with features and the user interface.

Strategy: *Separate components and modules along dimensions of concern.*

Follow the principle of separation of concerns to incorporate flexibility to accommodate change in the module view. Separate or decompose modules along important dimensions of concern, including processing, communication, control, data, and user interface aspects of the software design. For example, when designing an acquisition procedure, you want to separate the processing aspects from the control aspects. This provides the possibility of using the processing modules in other acquisition procedures or in contexts that cannot be foreseen at this time. Application of this strategy will also allow you to allocate and trace the requirements to the design elements. When the requirements change, it will be easier to reuse the existing framework and to plug in new components to get a new solution more quickly.

Strategy: *Encapsulate features into separate components.*

To isolate the effects of change to product features, organize related product features into separate components (for example, movement of the probe, image processing, connectivity to the network).

Strategy: *Decouple the user interaction model.*

To isolate the effects of change in the way the user interacts with the system, decouple the user interaction model from the rest of the applications.

Related Strategies

See also *Encapsulate general-purpose hardware* (issue, General-Purpose and Domain-Specific Hardware).

Table 3.5 summarizes the strategies we have developed so far. We will continue to analyze factors and to develop strategies in the next four chapters, which describe the design of the four architecture views. Table 3.5 also summarizes the factors and strategies not yet presented; they are included in the shaded rows.

Issue	Influencing Factors	Applicable Strategy
Aggressive Schedule	O4.1, O5.1, O1.1, O4.2, P7.2	*Reuse existing in-house, domain-specific components.*
		Buy rather than build.
		Make it easy to add or remove features.
Skill Deficiencies	O2.3, O2.4	*Avoid the use of multiple threads.*
		Encapsulate multiprocess support facilities.
Changes in General-Purpose and Domain-Specific Hardware	T1.1, T2.1, T2.2	*Encapsulate domain-specific hardware.*
		Encapsulate general-purpose hardware.
Changes in Software Technology	T3, T5	*Use standards.*
		Develop product-specific interfaces to external components.
Resource Limitations	T1.3, T3.2, T3.4	*Limit the number of active processes.*
		Use dynamic interprocess communication connections.
Easy Addition and Removal of Features	O4.1, O4.2, P1, P2.1	*Separate components and modules along dimensions of concern.*
		Encapsulate features into separate components.
		Decouple the user interaction model.

Table 3.5. Summary of Strategies

Issue	Influencing Factors	Applicable Strategy
Easy Addition and Removal of Acquisition Procedures and Processing Algorithms	O4.1, O4.2, P1.1, P1.2, P1.3	*Use a flexible pipeline model for image processing.*
		Introduce components for acquisition and image processing.
		Encapsulate domain-specific image data.
High Throughput	P7.1, T1.2, T3.2, T3.3, O2.3, O2.4	*Map independent threads of control to processes.*
		Use an additional CPU.
Real-Time Acquisition Performance	T1, T3.1, T3.2, T3.3, T3.4, T2.1, P3.1, P3.2	*Separate time-critical from nontime-critical components.*
		Develop guidelines for module behavior.
		Use flexible allocation of modules to processes.
		Use rate monotonic analysis to predict performance.
		Use shared memory to communicate between pipeline stages.
Implementation of Recovery	P5.4	*Introduce a recovery mode of operation.*
		Make all data at the point of recovery persistent and accessible.
Implementation of Diagnostics	P5.1, P5.2, P5.3	*Define an error-handling policy.*
		Reduce the effort for error handling.
		Encapsulate diagnostic components.
		Use standard logging services.

Table 3.5. Summary of Strategies *(continued)*

Issue	Influencing Factors	Applicable Strategy
Architectural Integrity	O3.2, T3.5, T5.6	*Preserve module view hierarchies.*
		Separate organization of public interface components.
Concurrent Development Tasks	O4.3, O3.4	*Separate organization of deployment components from source components.*
		Preserve execution view.
		Use phased development.
		Release layers through static libraries.
Limited Availability of Probe Prototypes	O3.2, O3.3, O3.4, O4.3	*Develop an off-line probe simulator with an appropriate abstraction.*
		Use a flexible build procedure.
Multiple Development and Target Platforms	T1.1, T1.2, O3.1, O3.2	*Separate and encapsulate code dependent on the target platform.*

Table 3.5. Summary of Strategies *(continued)*

3.8 Global Analysis Summary

The purpose of the global analysis is to identify the important factors that affect the architecture, analyze their changeability and impact, and develop strategies to guide the architecture design. Three kinds of influencing factors are considered during global analysis:

1. Organizational factors that constrain the design choices
2. Technological factors that are embedded or embodied in the product
3. Product factors that include functional features and qualities of the product

Typical categories of these three kinds of factors are summarized in Table 3.6.

Global analysis is an activity that should be performed as part of the design of each of the four architecture views. The issue cards produced during global analysis are used in the central design tasks of each view. Decisions made when performing the central design tasks could also cause a return to the global analysis to add or to modify factors, issues, or strategies.

Organizational Factors	Technological Factors	Product Factors
O1: Management O2: Staff skills, interests, strengths, weaknesses O3: Process and development environment O4: Development schedule O5: Development budget	T1: General-purpose hardware T2: Domain-specific hardware T3: Software technology T4: Architecture technology T5: Standards	P1: Functional features P2: User interface P3: Performance P4: Dependability P5: Failure detection, reporting, recovery P6: Service P7: Product cost

Table 3.6. Typical Categories of Influencing Factors

Global analysis has two activities, each with three steps. During the first activity, analyze factors, you must do the following:

1. Identify and describe the factors.
2. Characterize their flexibility and changeability.
3. Analyze their impact.

The result of this activity is a factor table for each of the three types of factors.

During the second activity, develop strategies, you must do the following:

1. Identify issues and influencing factors.
2. For each issue, develop a solution and specific strategies.
3. For each issue, identify related strategies.

The result of this activity is an issue card for each issue you identify. It is also useful to summarize the strategies in a table that cross-indexes issues, influencing factors, and strategies.

When global analysis is complete (at the end of architecture design), you will have characterized the important influencing factors and developed strategies for ensuring buildability, implementation, and changeability of the product and its architecture.

Additional Reading

Meszaros and Doble (1997) capture some of the best practices of pattern writing to make them more understandable and usable. This chapter follows some of the recommendations made in this paper. Coplein (1995) has more information on organization or process patterns.

Rechtin and Maier (1997) describe many important strategies for designing a system architecture that are applicable to software architecture. Davis (1995) describes 201 general strategies for software development. These strategies are useful as a starting point for selecting your own, either using them directly or adapting them for the needs of your system.

Dorofee et al. (1996) describe a method for continuous risk management for the entire development spectrum and a taxonomy of factors. Risk is defined as "the possibility of suffering loss." If the event is a certainty, it is not a risk but a problem. Managing risk is one aspect of global analysis. Other aspects of global analysis include identifying factors that impact the software architecture, the resulting problems and issues, and their associated strategies that guide design decisions.

Requirements engineering (Davis, 1993) addresses similar concerns as analyzing product factors that influence the architecture. Requirements engineering analysis tasks include grouping and organization, dependency and conflict analysis, and requirements quantification. Global analysis broadens this specialty to include organizational and technology factors to ensure the system is buildable and uses the results of the analysis to develop strategies that guide design.