A02-1. *Siemens Approach to Architecture Design*

2014

Sungwon Kang

Siemens 4 Views

- Conceptual View
  - Describes the system in terms of its major design elements and the relationships among them

    E.g.) Basic design elements:
      - Communicating objects
      - Components and connectors
- Module View
  - Decompose the system and the partitioning modules into layers
  - Partitions work among programmers
- Execution View
  - Allocate functional components to runtime entities
  - Show communication, coordination and synchronization
- Code View
  - Organize the source code into object code, libraries and binaries, then in turn into versions, files and directories

Specific Concerns of Conceptual View

- How does the system fulfill the requirements?
- How are the commercial off-the-shelf(COTS) components to be integrated and how do they interact (at the functional level) with the rest of the system?
- How is domain-specific hardware and/or software incorporated into the system?
- How is functionality partitioned into product releases?
- How are product lines supported?
- How can the impact of changes in requirements or the domain be minimized?

Specific Concerns of Module View

- How is the product mapped to the software platform?
- What system support/services does it use, and exactly where?
- How can testing be supported?
- How can dependencies between modules be minimized?
- How can reuse of modules and subsystems be maximized?
- What techniques can be used to insulate the product from changes in COTS software, in the software platform, or changes to standards?

Specific Concerns of Execution View

- How does the system meet its performance, recovery, and reconfiguration requirements?
- How can one balance resource usage (for example, load balancing)?
- How can one achieve the necessary concurrency, replication, and distribution without adding too much complexity to the control algorithms?
- How can the impact of changes in the runtime platform be minimized?

Specific Concerns of Code View

- How can the time and effort for product upgrades be reduced?
- How should product versions and releases be managed?
- How can build time be reduced?
- What tools are needed to support the development environment?
- How are integration and testing supported?

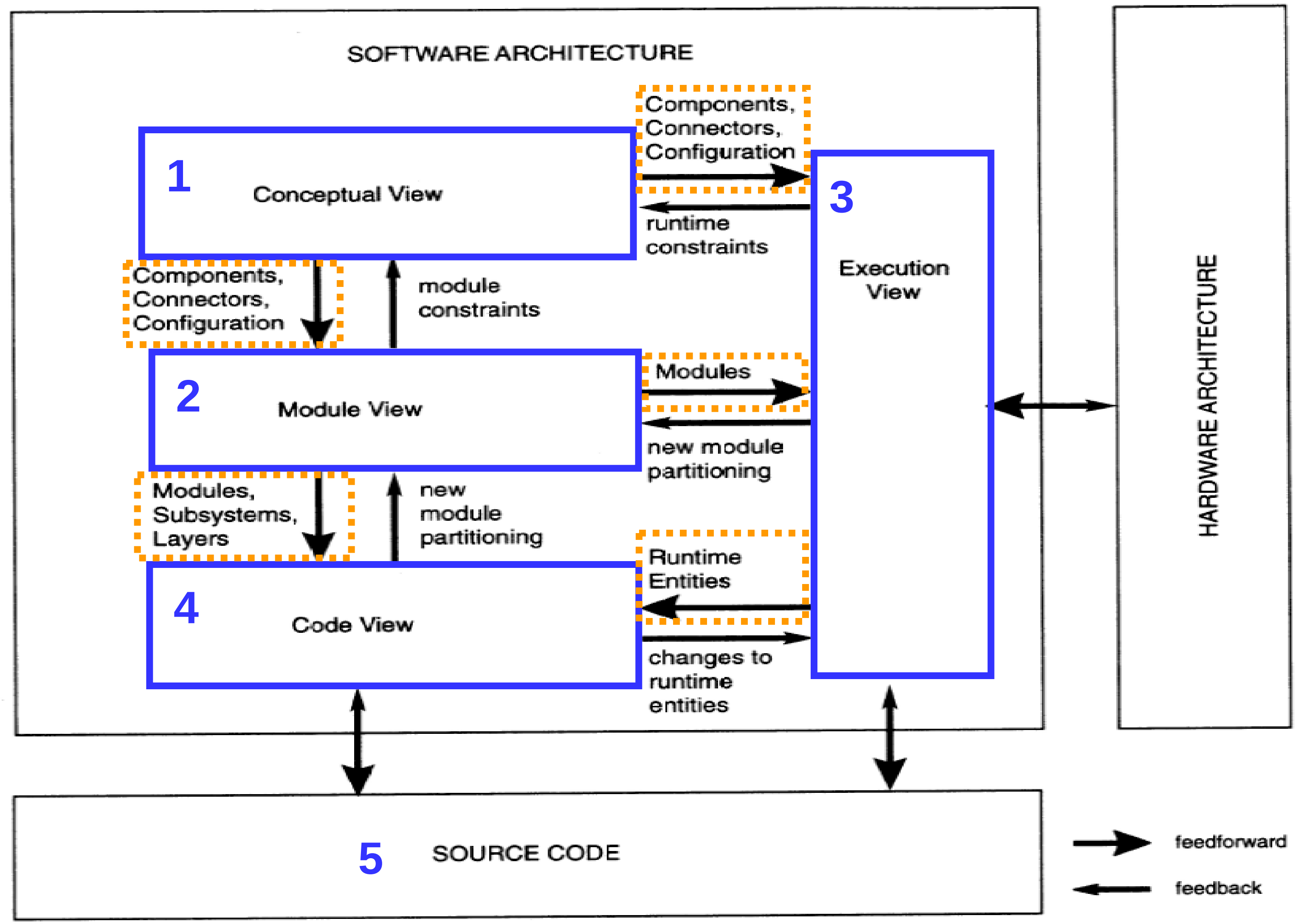


Figure 1.2. The four views of software architecture

System Requirements

IS 2000 System (Imaging Solution)

2.1 System Overview

IS2000's hardware configuration is shown in Figure 2.1. The basic functionality of the product is to capture an image using the camera on the probe. The source of the image could be from visible light, which is one form of electromagnetic radiation, from another frequency in the electromagnetic spectrum, from heat, or from some other source of radiation. To supplement the images, sensor readings provide additional information, such as the distance of the camera to the object being imaged. To make the images, IS2000 has to acquire this raw data and convert it into sensor readings and images suitable for viewing.

The IS2000 user may wish to take a single image, a series of images at different time intervals, or to create a motion picture. The user may wish to keep the camera stationary or move it to take images from different angles. Each of these options is an example of an *acquisition procedure*. The system has a set of built-in acquisition procedures, which can be customized by setting parameters before or during the acquisition. The system also allows the user to define a new acquisition procedure.

When the acquisition procedure is underway, the user is able to monitor it by viewing the raw data as it is transformed into a raw image. After the raw images are acquired, the user can perform a number of processing operations to enhance the images for viewing. The user can elect to transfer them to a remote imaging system for remote viewing or processing.

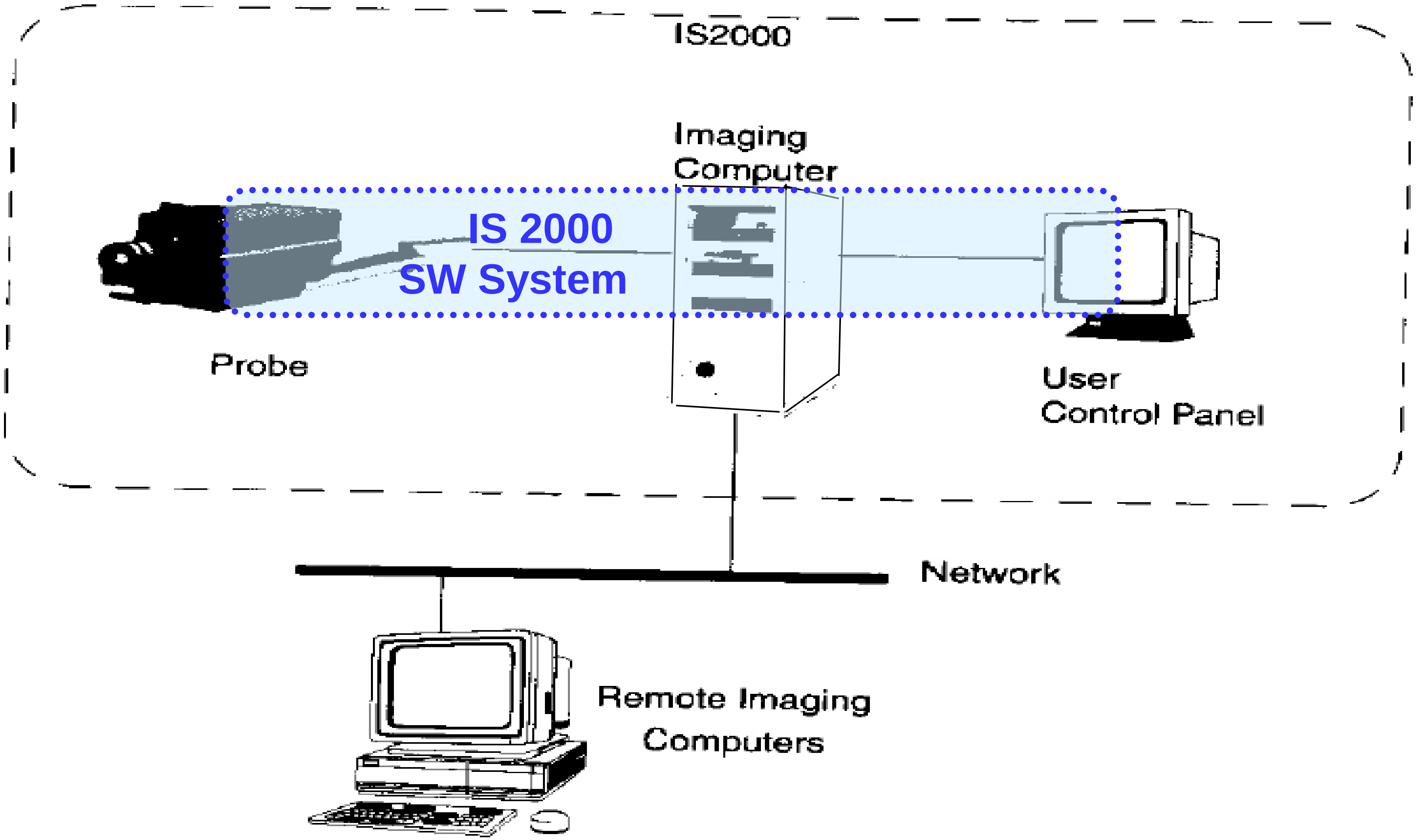


Figure 2.1. Hardware configuration of IS2000

2.2 Product Features

IS2000 provides a range of image acquisition and processing operations on two-dimensional and three-dimensional images, data storage for the images, and network access to the images. Its key marketing features are the following:

- IS2000 has a user-friendly operator environment.
- IS2000 has a comprehensive catalog of built-in acquisition procedures.
- The user can define custom acquisition procedures.
- The throughput of image acquisition is 50 percent higher than for previous products.
- Image display can be as fast as the maximum hardware speed.
- At runtime the user can make a trade-off between acquisition speed and image quality.
- IS2000 is designed for easy upgrade to new platforms.
- IS2000 has open platform connectivity to on-site or remote viewing and image postprocessing workstations.
- IS2000 can be connected to peripherals, including printers and digital imagers.

A single acquisition control panel is provided with each IS2000 unit. Additional independent viewing stations are also available.

2.3 System Interactions

IS2000 interacts with three things in its environment:

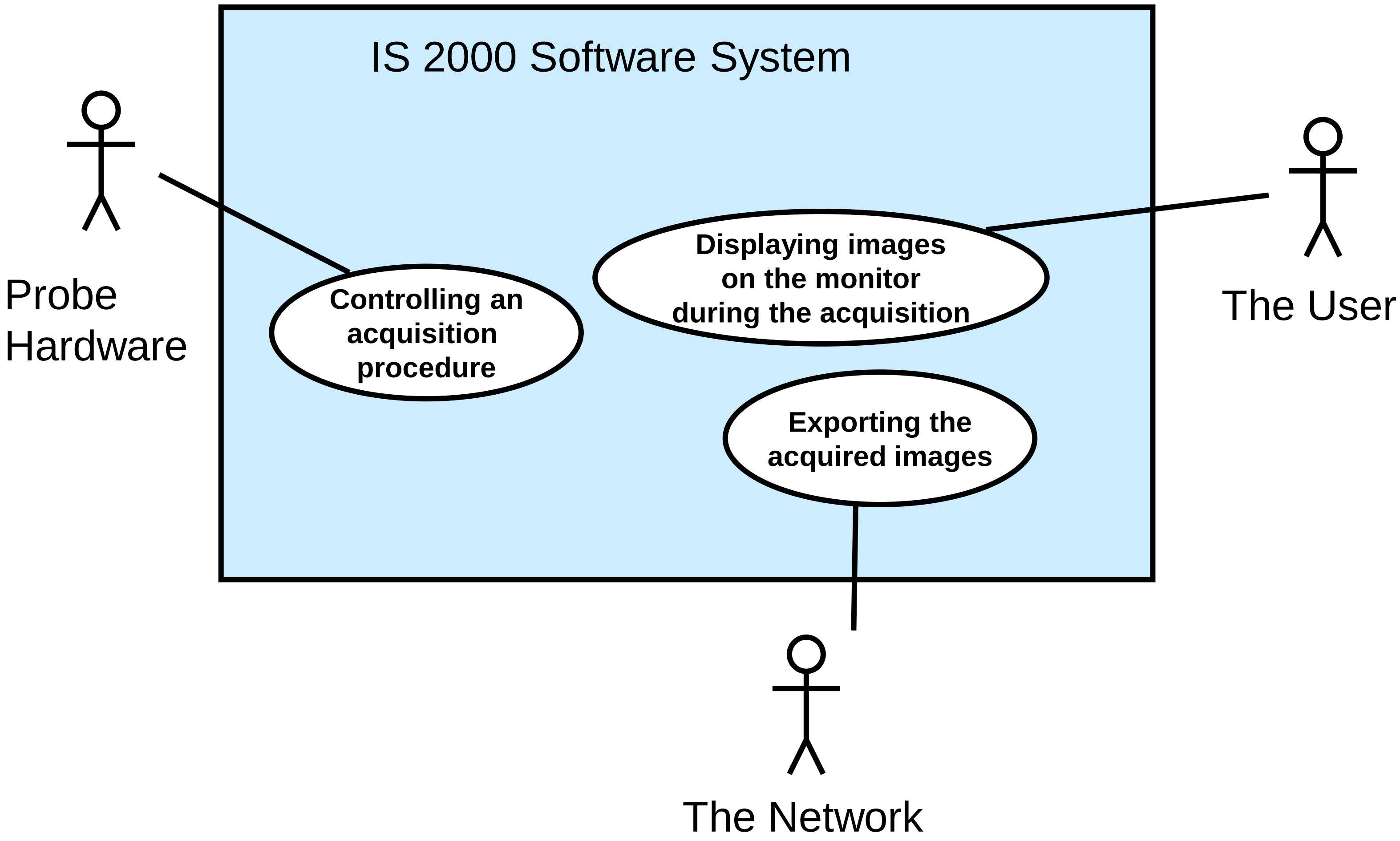
1. The user controlling the acquisition
2. The object being probed
3. The network connecting the system to other viewing stations

Context Diagram?
See Next Slide

IS2000 has a corresponding piece of hardware for each of these interactions: the user control panel, the probe hardware, and the network connector. These in turn each have an interface to IS2000's software, which runs on the imaging computer:

- The user control panel interface is used for setting the parameters for an acquisition, monitoring the acquisition, and selecting images for export.
- The probe hardware interface is used for controlling and receiving data from the probe hardware.
- The network connector interface is used for exporting image data over the network.

Exercise - Context Diagram



2.4 The Future of IS2000 (Change Aspect)

IS2000 must be designed to be extensible, maintainable, and portable. Its design must be flexible enough to accommodate certain expected changes. The product requirements may change somewhat during development and certainly will change over the lifetime of the product. The physical characteristics of the probe/camera 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 want more efficient ways of using it. As the processing power of the system increases, more and more work that was the responsibility of the users will shift to the system.

Other product features will also likely change. Over time, the built-in acquisition procedures will evolve. Image processing is constantly being improved, and new kinds of 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.

Architecture Design Steps

Step 0) Initial Global Analysis

- Derive Factors (Product, Organizational, Technological) (≈ architecture drivers)
- Derive Issues and solution strategies (≈ architecture problem analysis)

Steps 1-4) For each view (i.e. Conceptual, Module, Execution, Code),

- Global Analysis
  - Derive additional factors
  - Derive additional issues and solution strategies
- Central Design
- Final Design

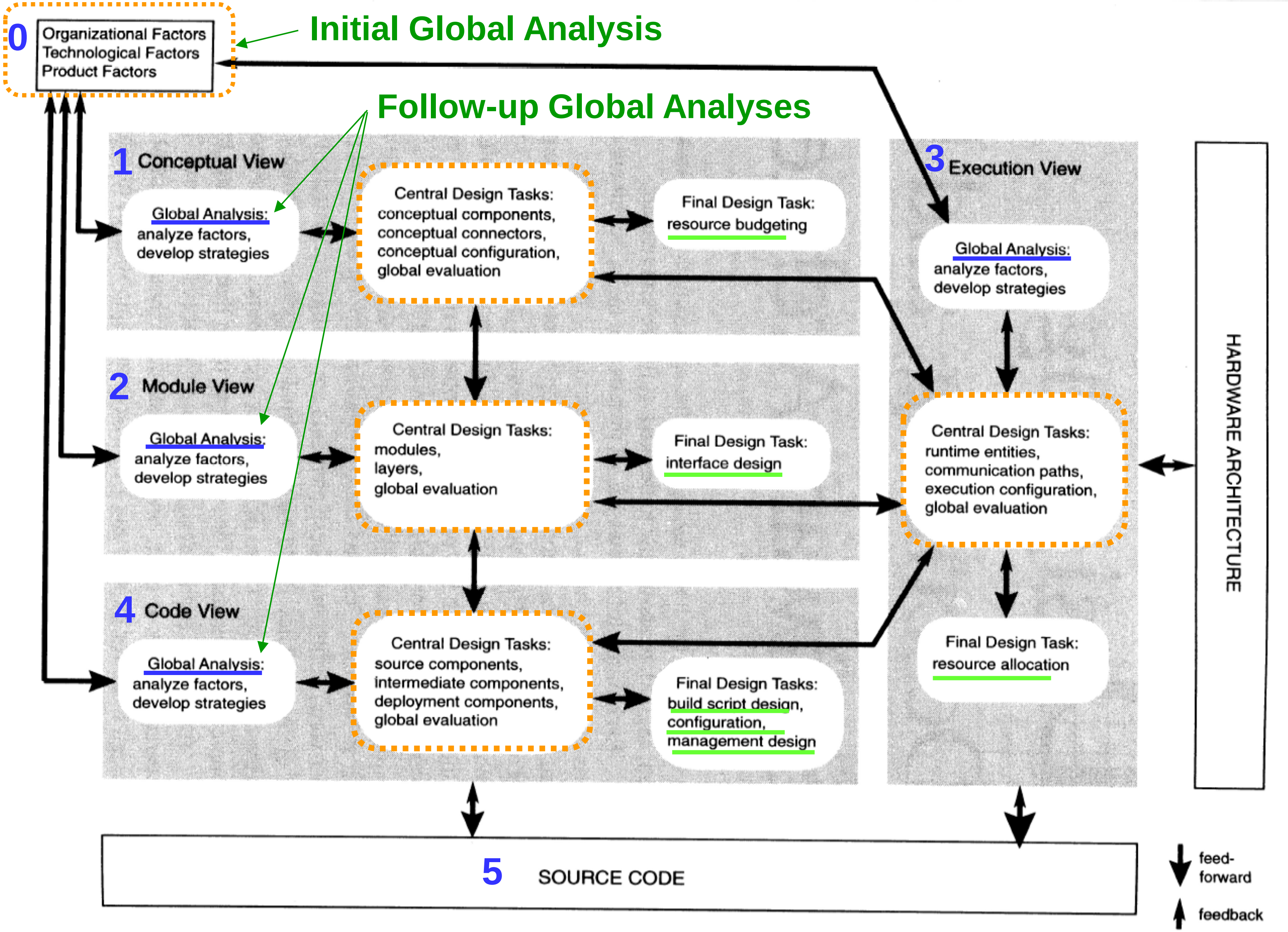
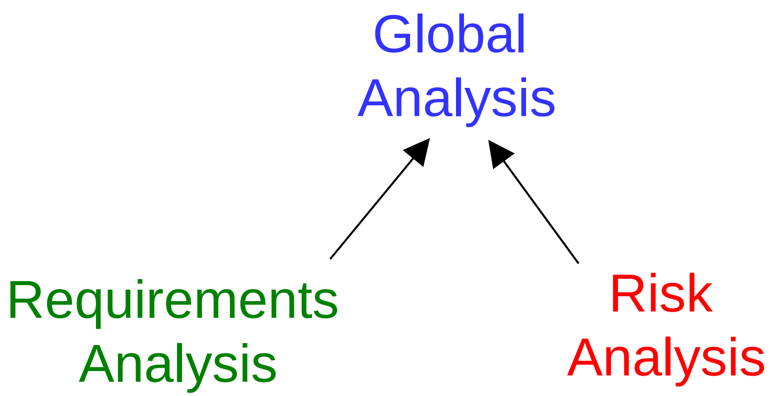


Figure II.1. Overview of the design tasks for the four views

Initial Global Analysis

Global Analysis

- A systematic way of
  - Identifying the factors that affect architecture, i.e. that
    - have global influence
    - could change during development
    - are difficult to satisfy or
    - you have little experience with
  - Developing strategies to handle them



Typical Product Factors

P1: Functional Features Functional features	**P5: Failure detection, reporting, recovery** Error classification Error logging Diagnostics Recovery
P2: User Interface User interaction model User interface features	
P3: Performance Acquisition performance Sensor data rate Start-up and shutdown times Recovery time	**P6: Service** Service features Software installation and upgrade Maintenance of domain-specific hardware Software testing Maintenance of software
P4: Dependability Availability Reliability Safety	**P7: Product Cost** Hardware budget Software licensing budget

Typical Organizational Factors

O1: Management Build vs. buy Schedule vs. functionality Environment Business goals	**O3: Process and development environment** Development platform Development process and tools CM process and tools Production process and tools Testing process and tools Release process and tools
O2: Staff skills, interests, strengths, weaknesses Application domain Software design Specialized implementation techniques Specialized analysis techniques	**O4: Development schedule** Time-to-market Delivery of features Release schedule
	O5: Development budget Head count Cost of development tools

Typical Technological Factors

T1:General-purpose hardware Processors Network Memory Disk	**T4: Architecture technology** Architecture styles Architecture patterns and frameworks Domain-specific or reference architectures ADLs Product line technologies
T2: Domain-specific hardware Probe hardware Probe network	
T3: Software technology OS UI Software components Implementation language Design patterns Frameworks	**T5: Standards** OS interface DB Data formats Communication Algorithms and techniques Coding conventions

Product Factors (IS 2000)

Requirements Product Factor	Severity and changeability Flexibility and Changeability	Impact analysis 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)*

Organizational Factors(IS 2000)

Constraints | **Severity and changeability** | **Impact analysis**

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)*

Technological Factors (IS 2000)

Constraints Technological Factor	Severity and changeability Flexibility and Changeability	Impact analysis 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)*

Issues

- Something we need to take care of

- An issue may be
  - Limitations or constraints imposed by factors

    E.g. aggressive schedule
  - The need to reduce the impact of changeability of factors.

    E.g. Reduce porting cost
  - The difficulty in satisfying product factors

    E.g. High throughput requirement may overload CPU
  - The need to have a common solution to global requirements

    E.g. Error handling and recovery

Issue 1

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.*** S1A

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.*** S1B

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.*** S1C

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.

Issue 2

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.* S2A

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.* S2B

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.

Issue 3

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.*** S3A

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.*** S3B

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.

Issue 4

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.*** S4A

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.*** S1B

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).

Issue 6

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.*** S6A

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.*** S6B

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).

S6C

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).

Summary of Issues and Strategies

Issue	Influencing Factors	Applicable Strategy
Aggressive Schedule Issue 1	O4.1, O5.1, O1.1, O4.2, P7.2	*Reuse existing in-house, domain-specific components.* S1A
		Buy rather than build. S1B
		Make it easy to add or remove features. S1C
Skill Deficiencies Issue 2	O2.3, O2.4	*Avoid the use of multiple threads.* S2A
		Encapsulate multiprocess support facilities.
Changes in General-Purpose and Domain-Specific Hardware Issue 3	T1.1, T2.1, T2.2	*Encapsulate domain-specific hardware.*
		Encapsulate general-purpose hardware.
Changes in Software Technology Issue 4	T3, T5	*Use standards.*
		Develop product-specific interfaces to external components.
Resource Limitations Issue 5	T1.3, T3.2, T3.4	*Limit the number of active processes.*
		Use dynamic interprocess communication connections.
Easy Addition and Removal of Features Issue 6	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

Factors to be addressed later

Factors to be addressed later

Issue	Influencing Factors	Applicable Strategy
Easy Addition and Removal of Acquisition Procedures and Processing Algorithms **Issue 7**	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 **Issue 8**	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 **Issue 9**	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 **Issue 10**	P5.4	*Introduce a recovery mode of operation.*
		Make all data at the point of recovery persistent and accessible.
Implementation of Diagnostics **Issue 11**	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)*

Factors to be addressed later

Issue	Influencing Factors	Applicable Strategy
Architectural Integrity Issue 12	O3.2, T3.5, T5.6	*Preserve module view hierarchies.*
		Separate organization of public interface components.
Concurrent Development Tasks Issue 13	O4.2, 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 Issue 14	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 Issue 15	T1.1, T1.2, O3.1, O3.2	*Separate and encapsulate code dependent on the target platform.*

Table 3.5. Summary of Strategies *(continued)*

Questions?