in real life

Global Analysis

PA1410 Software Architecture and Quality

Mikael Svahnberg[1]

[1] Mikael.Svahnberg@bth.se
School of Computing
Blekinge Institute of Technology

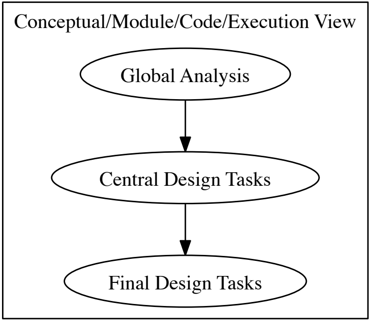
Outline

Background

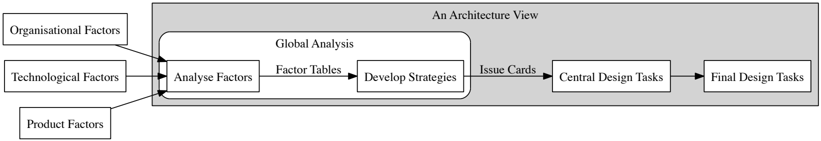
Example

Background on Global Analysis

- Hofmeister et al. (2000) “Applied Software Architecture”.
- A structured way of analysing a problem to find architecture drivers
- Overall development cycle:



Global Analysis Overview



Analyse Factors

Different types:

- Organisational Factors, e.g. development schedule, budget, in-house skills, development process
- Technological Factors, e.g. hardware, third party software, standards, architecture technology
- Product Factors, e.g. functional features, quality requirements

Steps

1. Identify and Describe the Factors
2. Characterise the Flexibility and Changeability of the Factors
3. Analyse the Impact of the Factors

Analyse Factors

Steps

1. Identify and Describe the Factors
2. Characterise the Flexibility and Changeability of the Factors
3. Analyse the Impact of the Factors

1. Identify and Describe the Factors

Does the factor have global influence (i.e., does it need to be dealt with by the [software] architecture)?

- Can the factor's influence be localised to one component in the design?
- During which stages of development is the factor important?
- Does the factor require any new expertise or skill?

Analyse Factors

Steps

1. Identify and Describe the Factors
2. Characterise the Flexibility and Changeability of the Factors
3. Analyse the Impact of the Factors

2. Characterise the Flexibility and Changeability of the Factor

What can be negotiated (flexibility), what may change (changeability)

F Is it possible to influence or change the factor to make your task of architecture development easier?

F In what way can you influence it?

F To what extent can you influence it?

C In what way could the factor change?

C How likely will it change during or after development?

C How often will it change?

C Will the factor be affected by changes in other factors?

Analyse Factors

Steps

1. Identify and Describe the Factors
2. Characterise the Flexibility and Changeability of the Factors
3. Analyse the Impact of the Factors

3. Analyse the Impact of the Factors

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

Develop Strategies

1. Identify Issues and Influencing Factors
2. Develop Solutions and Specific Strategies
3. Identify Related Strategies

Develop Strategies

1. Identify Issues and Influencing Factors
2. Develop Solutions and Specific Strategies
3. Identify Related Strategies

Step 1. Identify Issues and Influencing Factors

An Issue is something that needs to be addressed in the [software] architecture, often stemming from several factors and their changeability.

- An issue may arise from limitations or constraints imposed by factors. For example, agressive development schedule vs requirements overload.
- An issue may result from the need to reduce the impact of changeability of factors. For example, design the architecture to reduce the cost to porting to other operating systems.
- An issue may develop because of difficulties in satisfying product factors. For example, high throughput requirements may overload the CPU.
- An issue may arise from the need to have a common solution to global requirements. For example, error handling and recovery.

Develop Strategies

1. Identify Issues and Influencing Factors
2. Develop Solutions and Specific Strategies
3. Identify Related Strategies

Step 2. Develop Solutions and Specific Strategies

How you intend to address the issue. Design each strategy to be consistent with

- influencing factors
- desired/required changeability
- interactions with other factors

Address the following goals:

- Reduce or localise the factor's influence.
- Reduce the impact of the factor's changeability on the design and other factors.
- Reduce or localise required areas of expertise or skills.
- Reduce overall time and effort.

Outline

Background

Example

Example: Robot

Design an architecture for the application domain of mobile robots, for example the robot used in the latest mars expedition by NASA, a cleaning robot for sewerage pipes, oil pipeline maintenance robots or even a bomb disarming robot.

- Embedded Real-time system
- External Sensors and Actuators
  - Acquire input from sensors
  - control motion of wheels and moveable parts (actuators)
  - Plan the future path
- Factors that complicate:
  - Obstacles
  - Imperfect input
  - Power shortage
  - Mechanical limitations reduce accuracy of actuator movements
  - May manipulate hazardous materials
  - Unpredictable events may leave little time for responding

Design Considerations

The architecture must accomodate:

- deliberate as well as reactive behaviour
- uncertainty; re-planning and reacting
- dangers inherent in the robot’s operation and environment (fault tolerance, safety, performance)
- flexibility for the designer

Global Analysis / Analyse Factors

- Organisational factors
- Technical factors
- Product factors

Global Analysis / Analyse Factors

- Organisational factors
  - Leave these for now
- Technical factors
- Product factors

Global Analysis / Analyse Factors

- Organisational factors
- Technical factors
  - General-Purpose Hardware: Processors, Network, Memory, Disk
  - Domain-Specific Hardware: Sensor hardware, actuator hardware
  - Software Technology: Operating System, User Interface, Software Components, Implementation Language, Design Patterns, Frameworks
  - Architecture Technology: Architecture styles, patterns, frameworks, domain specific architectures, architecture description languages, product-line-technologies
  - Standards
- Product factors

Global Analysis / Analyse Factors

- Organisational factors
- Technical factors
- Product factors

- Functional Features
- User Interface
- Performance, Dependability (Availability, Reliability, Safety)
- Failure detection, reporting, recovery: Error classification, Error logging, diagnostics, recovery
- Service: Service features, software installation and upgrade, maintenance of hardware, software testing, software maintenance
- Product Cost: hardware budget, software licencing budget

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface		

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	New sensor; new UI component

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	New sensor; new UI component
User-defined set of Actuators		

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	New sensor; new UI component
User-defined set of Actuators	F: Ok to select from pre-defined	

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	New sensor; new UI component
User-defined set of Actuators	F: Ok to select from pre-defined	No need for open API for actuators / no need to sandbox actuator controllers

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	New sensor; new UI component
User-defined set of Actuators	F: Ok to select from pre-defined	No need for open API for actuators / no need to sandbox actuator controllers
Sensor Interpretation is based on multiple sensors		

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	New sensor; new UI component
User-defined set of Actuators	F: Ok to select from pre-defined	No need for open API for actuators / no need to sandbox actuator controllers
Sensor Interpretation is based on multiple sensors	None	

Analyse Factors (examples)

Note: Examples are not structured according to organisational, technological, or product factors.

Factor	Flexibility & Changeability	Impact
Networked User Interface	C: May extend to radio-controlled	New sensor; new UI component
User-defined set of Actuators	F: Ok to select from pre-defined	No need for open API for actuators / no need to sandbox actuator controllers
Sensor Interpretation is based on multiple sensors	None	-

Develop Strategies

1. Identify Issues and Influencing Factors
2. Develop Solutions and Specific Strategies
3. Identify Related Strategies

Develop Strategies

1. Identify Issues and Influencing Factors
2. Develop Solutions and Specific Strategies
3. Identify Related Strategies

Step 1. Identify Issues and Influencing Factors

An Issue is something that needs to be addressed in the [software] architecture, often stemming from several factors and their changeability.

- An issue may arise from limitations or constraints imposed by factors. For example, agressive development schedule vs requirements overload.
- An issue may result from the need to reduce the impact of changeability of factors. For example, design the architecture to reduce the cost to porting to other operating systems.
- An issue may develop because of difficulties in satisfying product factors. For example, high throughput requirements may overload the CPU.
- An issue may arise from the need to have a common solution to global requirements. For example, error handling and recovery.

Develop Strategies

1. Identify Issues and Influencing Factors
2. Develop Solutions and Specific Strategies
3. Identify Related Strategies

Step 2. Develop Solutions and Specific Strategies

How you intend to address the issue. Design each strategy to be consistent with

- influencing factors
- desired/required changeability
- interactions with other factors

Address the following goals:

- Reduce or localise the factor's influence.
- Reduce the impact of the factor's changeability on the design and other factors.
- Reduce or localise required areas of expertise or skills.
- Reduce overall time and effort.

Develop Strategies (examples)

Issues

- Mechanical limitations in actuators: accuracy of actuator movements are less than optimal.
  - Solution: Use sensors to continuously verify actuator movements
- Rough Environment: Communication with driver may not always work
  - Solution: Build plan based on driver's previous input
  - Solution: Buffer feedback for driver and re-send until acknowledged
  - Solution: Verify sanity of input against plan and previous input
- Unknown number of sensors/actuators may slow down main control loop
  - Solution: Parallelize sensor/actuator processing

Conceptual View

Note: Creating viewpoints is not the goal of this lecture; this is included as a reference to understand how to go from issues and strategies to a design in a traceable way.

- Which Issues are useful to create this view?
- Which Factors?

Conceptual View

Note: Creating viewpoints is not the goal of this lecture; this is included as a reference to understand how to go from issues and strategies to a design in a traceable way.

- Which Issues are useful to create this view?
  - Mechanical limitations in actuators: requires synchronisation between actuators and sensors → interaction between components
  - Rough Environment: new components (PlanCreator, FeedbackBuffer, InputSanityChecker)
  - *Not* Unknown number of sensors and actuators
- Which Factors?

Conceptual View

Note: Creating viewpoints is not the goal of this lecture; this is included as a reference to understand how to go from issues and strategies to a design in a traceable way.

- Which Issues are useful to create this view?
  - Mechanical limitations in actuators: requires synchronisation between actuators and sensors → interaction between components
  - Rough Environment: new components (PlanCreator, FeedbackBuffer, InputSanityChecker)
  - *Not* Unknown number of sensors and actuators

Why not?

Parallelization does not *per se* create any new components; it simply allocates the components to threads, processes, and processors. You may create a new coordination component but we will have this anyway in order to interpret sensor input based on multiple sensors.

- Which Factors?

Conceptual View

Note: Creating viewpoints is not the goal of this lecture; this is included as a reference to understand how to go from issues and strategies to a design in a traceable way.

- Which Issues are useful to create this view?
- Which Factors?
  - Networked User Interface: new components (UserIO, UserInterface), interactions with components
  - Sensor Interpretation is based on multiple sensors: new components (SensorInputInterpreter)
  - *Not* User-defined set of Actuators

Conceptual View

Note: Creating viewpoints is not the goal of this lecture; this is included as a reference to understand how to go from issues and strategies to a design in a traceable way.

- Which Issues are useful to create this view?
- Which Factors?
  - Networked User Interface: new components (UserIO, UserInterface), interactions with components
  - Sensor Interpretation is based on multiple sensors: new components (SensorInputInterpreter)
  - *Not* User-defined set of Actuators

Why not?

Does not create any new components, it is just a configuration item that can be dealt with by any number of variability realisation techniques[a]

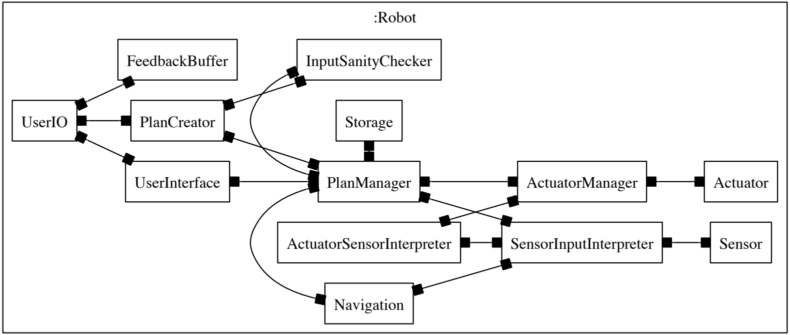
[a] M. Svahnberg, J. Van Gurp, and J. Bosch. A taxonomy of variability realization techniques. in *Software- Practice and Experience*, 35(8):705–754, 2005.

Conceptual View

- Identifying the issues and factors like this provides a rudimentary traceability.
- Next, you start designing. Start with a large box and add components as you need to provide the functionality of the system.
- Take care to address any factors you had regarding architectural styles, domain specific architectures, etc.
- For example, Alberto Elfes[1] provides a layered architecture for robots; These layers will probably be represented by (sets of) components in some way in the conceptual view, and even more so in the module view.

[1] Elfes. Sonar-Based Real-World Mapping and Navigation. IEEE Journal of Robotics and Automa- tion, no.3, 1987, pp. 249-265.

Conceptual View



Summary

- Global Analysis is a *tool* to help you identify architecture relevant concerns
- It provides a *structured approach*
- It provides basic *traceability* from your concerns to your actual architecture
- It does not help you to actually create a workable architecture; it just makes sure you do not forget anything.