Arkkitehtuuri- ja prosessimallit

Johannes Koskinen

Mallien lähde

Arkkitehtuurimallit

- Sulake-projekti 2008-2010
  - ”The aim of the Sulake project is to analyze and assess software architectures of existing embedded systems, especially in machine industry, and to identify good and harmful solutions in those architectures. Another central aim is to develop software architecture assessment methods and practices, especially in machine industry.”

Prosessimallit

- Ohjelmaturva-projekti 2010-

Esityksen rakenne

Arkkitehtuurimallit

- Rakenne
- Esimerkki arkkitehtuurimallista
- Kieli ja näin se koottiin

Prosessimallit

- Prosessimallien eri tasot
- Prosessimalli prosessimallille
  - ”Näin luot uuden mallin”
- Rakenne
- Esimerkki prosessimallista

Mallit (Patterns)

Pattern

- ”*a repeated decorative design*” (dict.)
- ”*(esineen ym.) malli, tyyppi, rakenne, muoto*” (MOT)
- ”*malli, tapa, tyyli (käyttäytymisestä ym.)*” (MOT)

”**(suunnittelu)malli** tarkoittaa yleistä tapaa jonkin usein esiintyvän ongelman ratkaisemiseksi.” (Wikipedia)

Arkkitehtuurimallit (rakenne)

Mallin määrittelevät peruskomponentit

- Nimi
  - Mallin kuvaava nimi
- Asiayhteys (*context*)
  - Missä yhteydessä mallia voidaan käyttää?
- Ongelma (*problem*)
  - Minkä ongelman malli ratkaisee? How to…
- Vaikutusvoimat (*forces*)
  - Miksi tätä mallia pitäisi käyttää ongelman ratkaisuun?

Arkkitehtuurimallit (rakenne)

- Ratkaisu (*solution*)
  - Mitä täytyy tehdä, jotta ongelma ratkeaisi?
- Seuraukset (*consequences*)
  - Mallin käytön positiiviset ja negatiiviset vaikutukset
- Lopputulos (*resulting context*)
  - Mikä on järjestelmän tila mallin käytön jälkeen?
- Sukulaismallit (*related patterns*)
  - Samanlaisia malleja, toisia malleja saman ongelman ratkaisuun, ratkaisuun käytettäviä malleja
- Käyttötapaus (*known usage*)
  - Esimerkki käytöstä

Esimerkki arkkitehtuurimallista

SAFE STATE

Esimerkki mallista (Safe State)

Context

A machine control system has been distributed using ISOLATE FUNCTIONALITIES pattern. A single node can malfunction and prevent the safe operation of the machine. When a machine with moving parts malfunctions, it can cause harm when let to move uncontrolled. However it might be problematic to determine what should be done when a malfunction occurs.

Problem

How to minimize the possibility that operator, machine or surroundings are harmed when some part of the machine malfunctions?

Esimerkki mallista (Safe State)

Forces

- *Response time*: Possible protective (in)action should be immediate.
- *Analyzability*: Functionality in safe state should be understandable and traceable.
- *Testability*: Emergency, fault, etc. situation should be testable.
- *Safety*: Operator, machine or surroundings should not be harmed.
- *Fault tolerance*: When a fault occurs, there should be mechanism that prevents further damage.
- *Recoverability*: It should be possible to continue operation after emergency situation occurs in predictable way.
- *Stability*: Safe state should be stable.

Esimerkki mallista (Safe State)

Solution

When a malfunction occurs harm should be prevented by entering a safe state. The safe state is device and functionality dependent and it is not necessarily the same as unpowered state. For example, the safe state of a brake controller is to maintain braking pressure in order to prevent movement to downhill. However, sometimes unpowered state is also the safe state. For example, if a forester harvester is feeding a log, the safe state is to stop movement. Determining the correct safe state can be challenging. However, each actuator or controller in the system should know at any given moment which is the safe state that should be entered if a malfunction occurs. Furthermore, the safe state may depend on type of the malfunction. Safe state should also be realized when the machine operator presses the emergency stop button.

Esimerkki mallista (Safe State)

When transition to safe state is taken, every controller enter predetermined safe state. Malfunction situation leading to the transition to safe state can be detected for example, by using solution provided by the HEARTBEAT pattern. Sometimes the node can also detect itself that it is malfunctioning. When device enters the safe state it should inform other nodes about the situation. This can be done for example, by sending an emergency message to the bus. Other nodes should react to this by entering their safe state. In this case, other nodes does not necessarily need to send another emergency message to the bus. Furthermore, sometimes it is required that multiple controllers enter the safe state in proper order so that the safe state can be achieved. This can make the solution more complex as nodes may need to send additional messages targeted to specific controllers.

Esimerkki mallista (Safe State)

Recovery from the safe state can be initiated by machine operator who first repairs the malfunctioning part and then restarts the system. Alternatively, service person should be able to start recovery from the safe state from maintenance user interface. Once the malfunction is solved, return to normal operation from safe state should be also handled in safe fashion. For example, in forest harvester if boom is holding a log in the air while in the safe state, the grappler should not open and drop the log when taking transition back to normal operation mode.

Esimerkki mallista (Safe State)

Consequences

+ The predetermined safe state can save lives, prevent accidents and material losses.
– The safe state is not necessarily the same in all cases and therefore it can be hard to determine which is the safe state. For example, in a machine that has frame controller which controls the body of the machine. Safe state of such controller can depend on the physical location of the machine. If the machine is in a slope, the controller application should not decide to release hydraulic pressure. However, on flat surface, the hydraulic pressure can be released.
– There can be dependency chains between controllers complicating the transition to safe mode.

Esimerkki mallista (Safe State)

Resulting Context

The result is a distributed and scalable machine control system where controllers can enter the devices into safe state.

Related Patterns

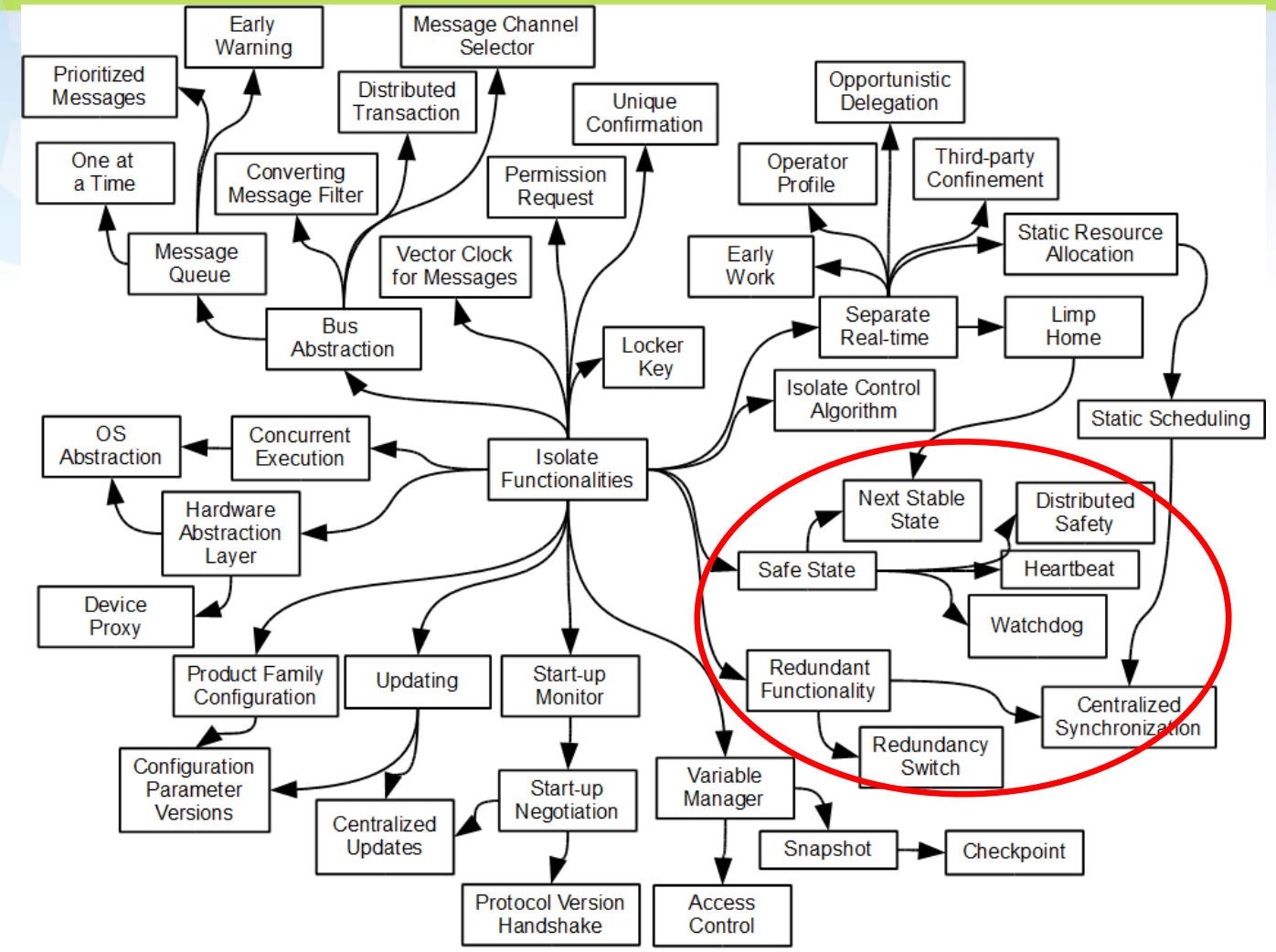
By using HEARTBEAT and WATCHDOG, situations where safe mode might be needed can be determined. Partially working machine can continue in degraded mode in some cases with LIMP HOME in order to reach state described by NEXT STABLE STATE.

Esimerkki mallista (Safe State)

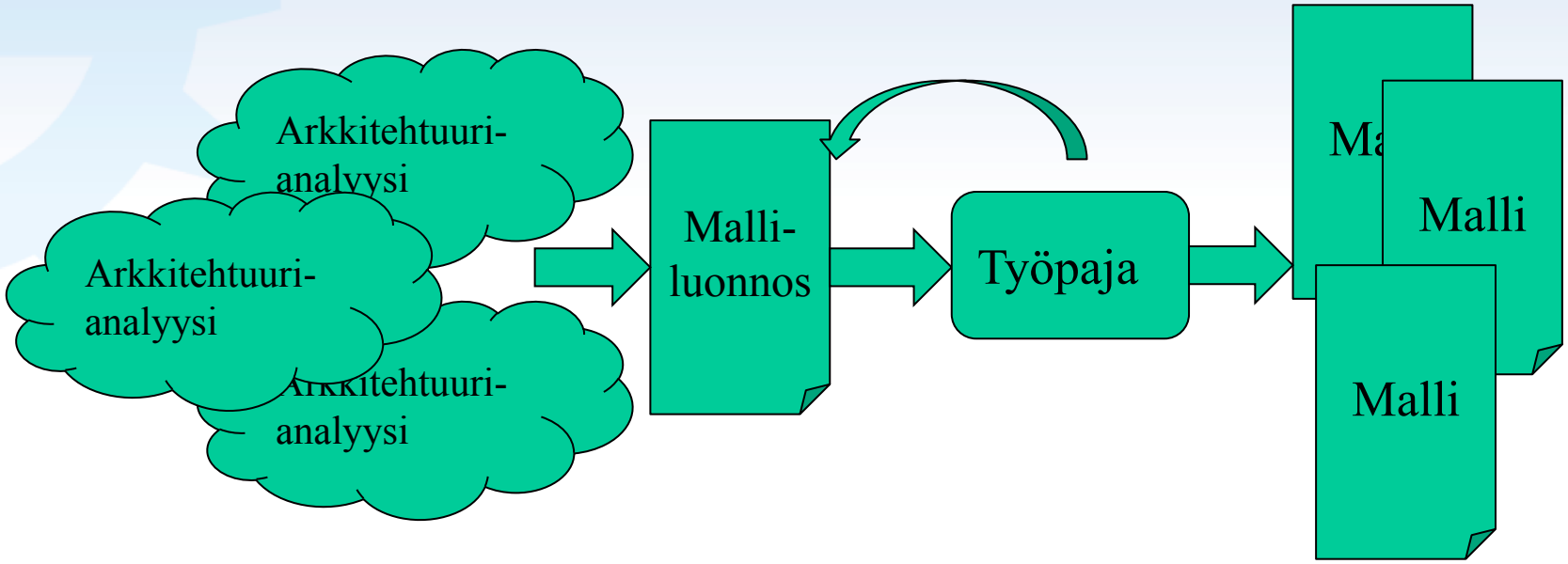
Known Usage

Forest harvester has grabbed a tree in its clamps and is delimbing the tree. For some reason, the harvester head is feeding the log in high speed in such an angle that the log is moving towards the cabin of the machine. The feed controller detects this potentially dangerous situation and enters its safe state and feeding is stopped. The feed controller sends an EMCY message to the CAN bus. Other nodes gets the message and enter their safe states. For example, boom controller does not release hydraulic pressure, but keeps the boom in the air. If the boom would be released it could also cause harm for the environment or the machine itself. The machine operator is informed about the situation on the screen in the cabin. Operator can evaluate the situation and may continue working by restarting the system.

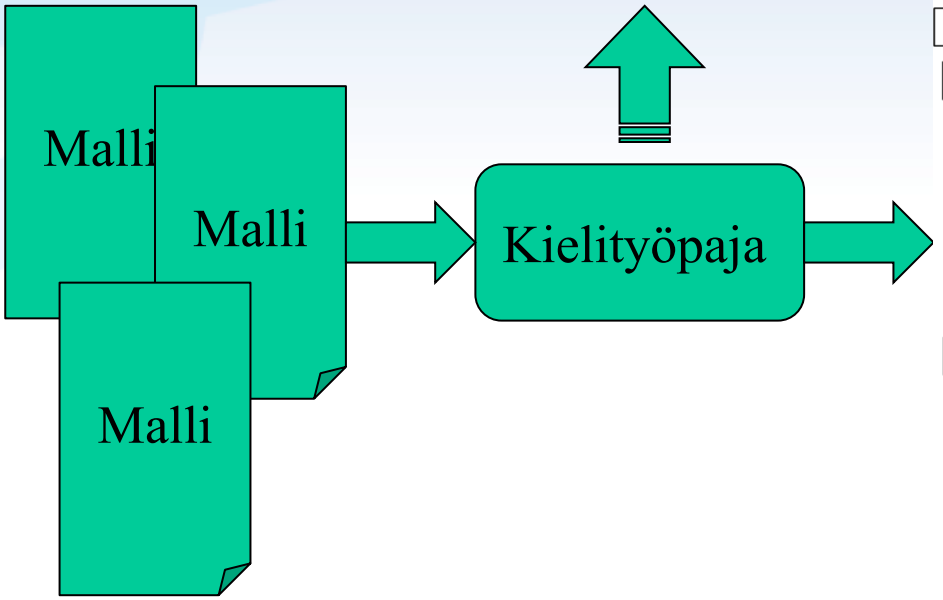
Kieli
Distributed Machine Control System

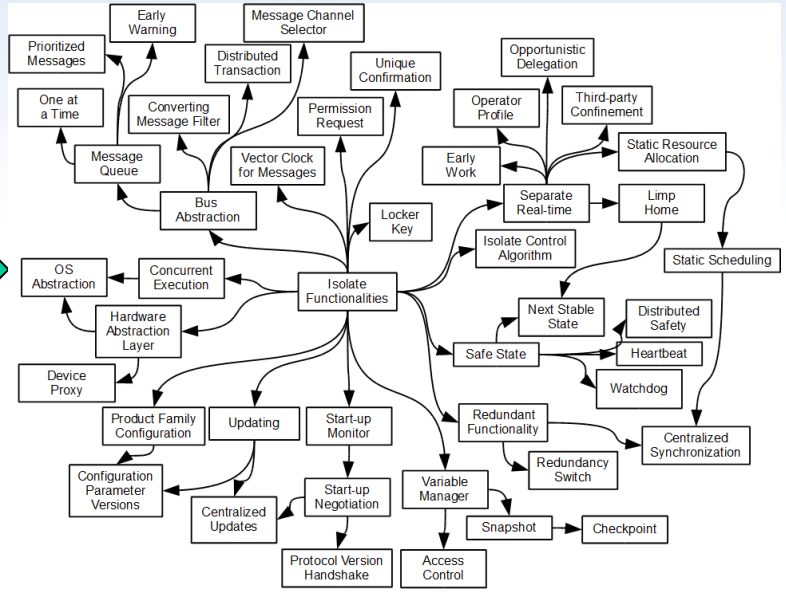


Näin kieli luotiin



Näin kieli luotiin





http://practise.cs.tut.fi/

KYSYMYKSIÄ?

Prosessimallit

Prosessi

”a series of actions or steps taken in order to achieve a particular end” (dict.)

”The patterns of activity within an organization” (Coplien)

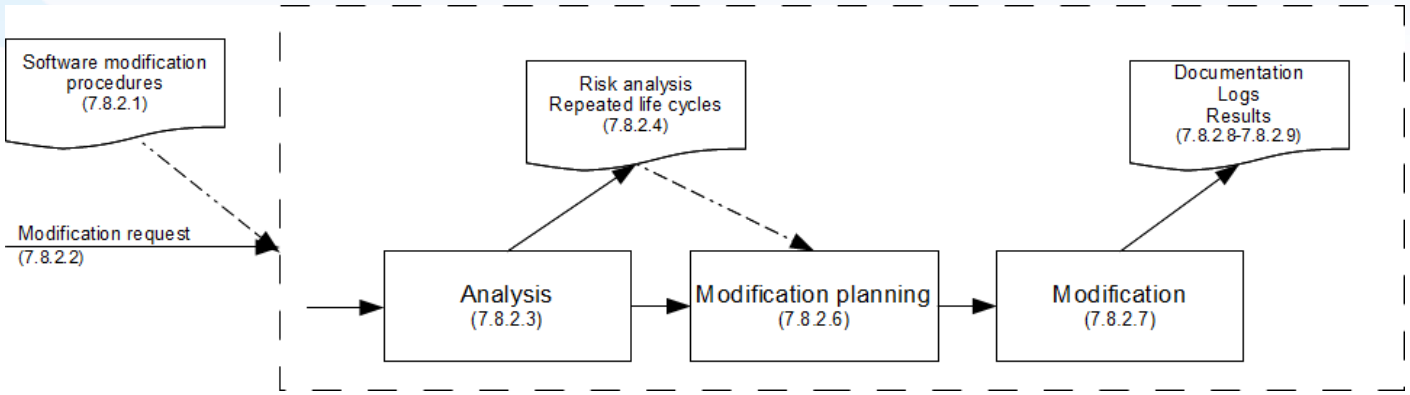
” Prosessi on sarja suoritettavia toimenpiteitä, jotka tuottavat määritellyn lopputuloksen.” (wikipedia)

Prosessimallit

”Process patterns can be defined as the set of activities, actions, work tasks or work products and similar related behavior followed in a software process lifecycle.” (wikipedia)

Tehtäväprosessimallit (Task Process Patterns)

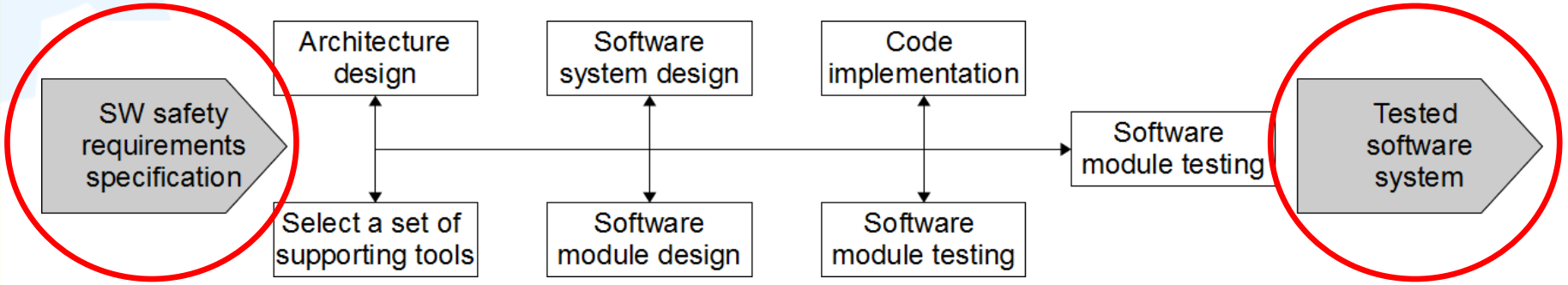
Nämä prosessimallit kertovat yksityiskohtaiset vaiheet jonkin tietyn tehtävän suorittamiseen.



2. **Muutospyyntö**. Muutosvaihe voidaan käynnistää vain hyväksytyllä muutospyynnöllä …
3. **Analyysi**. Analyysiä varten …

Työvaiheprosessimalli (Stage Process Patterns)

Tämän tyypin prosessimallit koostuvat yhden työvaiheen (yleensä) iteratiivisesti suoritettavista tehtävämalleista.

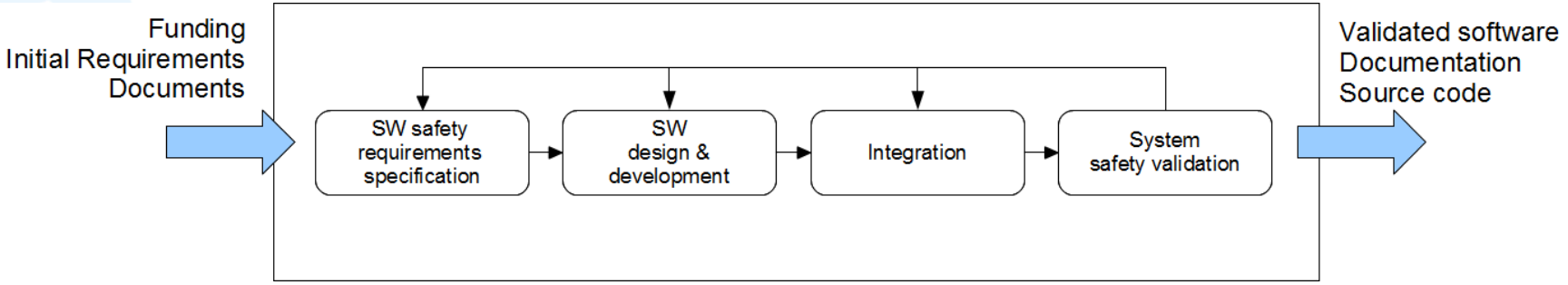


Software design and development

Työvaihemallit kertovat mitä tehtäviä tarvitaan yhden prosessin työvaiheen suorittamiseen.

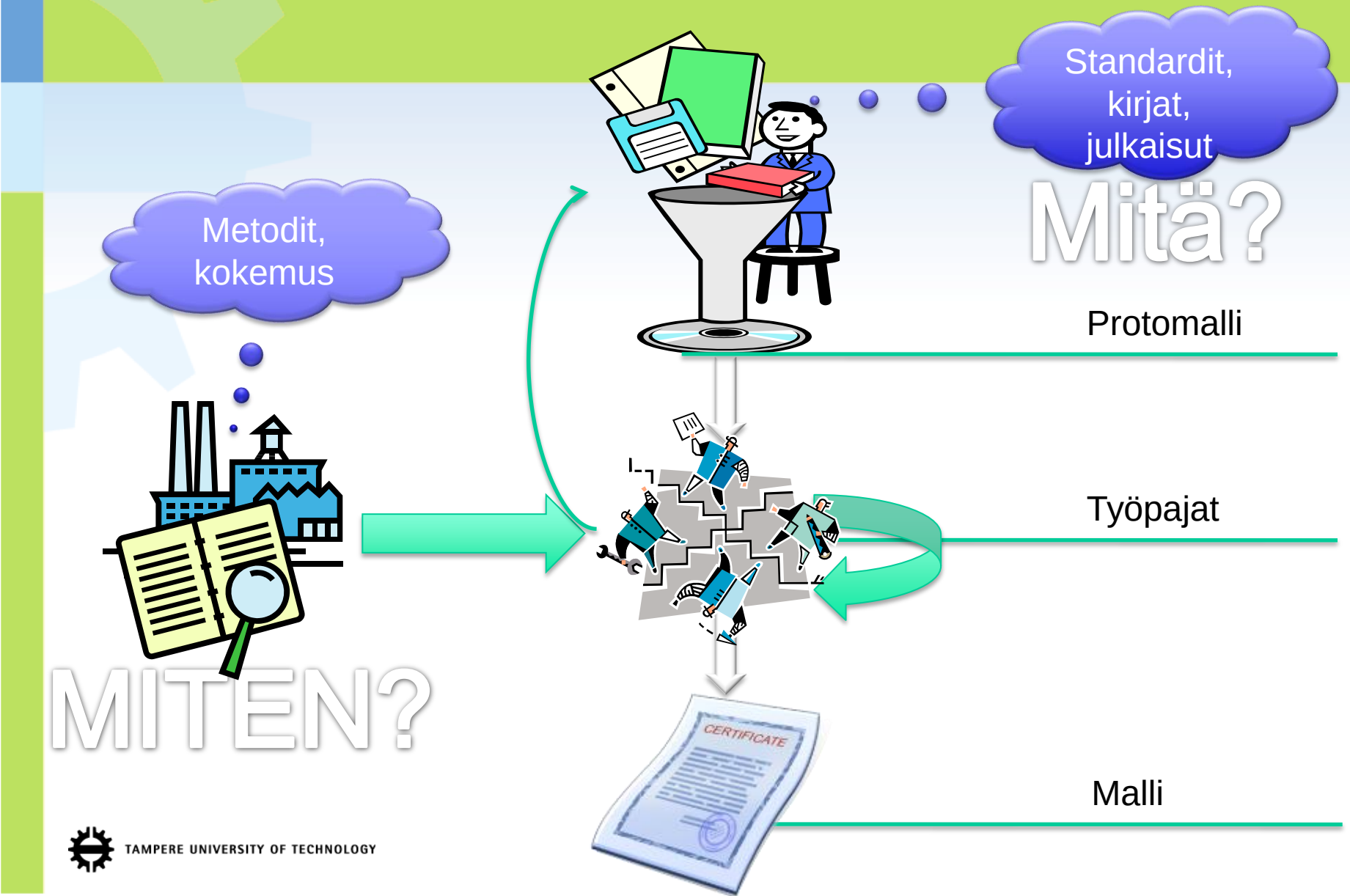
Vaiheprosessimallit (Phase Process Pattern)

Nämä prosessimallit kuvaavat eri työvaihemallien vuorovaikutuksen yhdessä projektin vaiheessa.



Vaihemallit suoritetaan järjestyksessä ja ne koostuvat työvaihemalleista, joita suoritetaan iteratiivisesti.

Prosessimallien löytäminen



Prosessimallien rakenne

Noudattaa arkkitehtuurimallin yleistä rakennetta

- Nimi
- Asiayhteys
- Ongelma
- Vaikutusvoimat
- Ratkaisu
- Lopputulos
- Sukulaismallit
- Standardiviitteet (IEC 61508)

Esimerkki tehtäväprosessimallista

SOFTWARE MODIFICATION

Esimerkki mallista (Software modification)

Context

There is an E/E/PE safety-related system where SOFTWARE VALIDATION has been applied. The software may need modifications, for example because of modifications to the overall safety requirements.

Problem

How to ensure that the required software systematic capability is sustained when the validated software is modified?

Forces

Modification should be complete and correct with respect to its requirements and unwanted behaviour caused by the modification shall be avoided. The design of the modified software should be verifiable and testable.

Esimerkki mallista (Software modification)

Abstract

Based on IEC 61508 standard, software cannot be maintained - it is always modified. When the validated software is modified (e.g. corrected or enhanced), it should be ensured that the functional safety for the E/E/PE safety-related systems is appropriate after the modification. Thus, the modification procedures shall be planned (MODIFICATION PLANNING) and made available. A modification can be initiated only by following the procedures specified during safety planning. In the Figure 1, the process for modification is illustrated briefly.

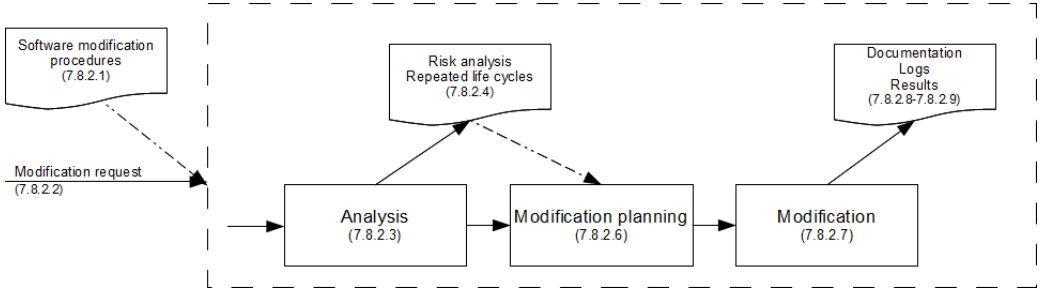


Fig. 1. Software Modification Process

Esimerkki mallista (Software modification)

First, it must be determined if a hazard and risk analysis is required and which software safety lifecycle phases will be repeated. The results are documented. It may even be necessary to implement a full hazard and risk analysis. The modifications and verification are planned in details. The planning should cover identification of required staff, detailed specification for the modifications, verification, and testing of the modifications (see IEC 61508-1 (6)). After the planning, the modifications are carried out as planned. Details of all modifications, including log files and re-verification and re-validation of data and results, shall be documented.

Esimerkki mallista (Software modification)

Solution

1. **Software modification procedures**. Software modification procedures are prepared and made available (IEC 61508-1 (7.16)).
2. **Modification request**. The modification phase is initiated only by the issue of an authorized request. The request should include the reasons for the change, proposed change and determined hazards that may be affected.
3. **Analysis**. An analysis is carried out on the impact of the proposed software modification to determine if a hazard and risk analysis is required and which software safety lifecycle phases will need to be repeated. The analysis is documented. All modifications which have an impact on the functional safety of the E/E/PE safety-related system cause a return to an appropriate phase of the software safety life-cycle.

Esimerkki mallista (Software modification)

All subsequent phases are carried out in accordance with the procedures specified for the phases.

4. **Risk analysis**. A full hazard and risk analysis is implemented if necessary. It may generate a need for different safety integrity levels than currently specified.
5. **Modification planning**. Safety planning is used to detail all subsequent activities. The safety planning for the modification may require involvement of domain experts. The planning should meet the requirements given in IEC 61508-1 (6).
6. **Modification**. After the planning, the modification is carried out as planned.
7. **Documentation**. Details of all modifications are documented.

Esimerkki mallista (Software modification)

8. **Logs**. Information (like log files) of all modifications are documented .

   The assessment of the required modification should be dependent on the results of the impact analysis and the software systematic capability.

Resulting Context

The E/E/PE safety-related system with modified software. All the details of the modification are documented for traceability.

Standard References

IEC 61508-3 (7.8) describes guidelines to corrections, enhancements or adaptations to the validated software. IEC 61508-1 (7.16) defines software modification procedures. IEC 61508-3 (Table A.8 and Table C.8) list techniques and properties used in various safety integrity levels.

KYSYMYKSIÄ?