

The Foss case - designing the S4000

Handling the coordination

Content:

- Foss Electric and the S4000 project
- Means for handling the coordination work
- A few examples
- Modeling needs

Purpose:

Illustrate the complexity of "real life" that we must cope with

An S4000 !

Foss Electric A/S

Foss Electric

- Instruments for measuring quality parameters of agricultural products, mostly dairy products
- The customers are laboratories, slaughterhouses, dairies, etc.
- 700 employees worldwide
- Large market-share (90%), export oriented
- Focus on:
 - a very high competence level
 - a matrix-organization
 - concurrent engineering

The S4000 project

- Analytical testing of raw milk (e.g. fat, lactose, protein, urea, citric acid)
- 200,000 lines of software. Configuration, control, and operation via a Windows user interface (Intel-based 486 PC build-in)
- Very high demands on speed and new facilities
- 8000 components: Cabinet, pipette unit, conveyor, PC, other hardware, flow-system, and measurement unit
- Project size up to 50 people at a time. Project (version 1) ran for 2 1/2 years

Main Competences:

- Mechanical design
- Electronical design
- Software design
- Chemical design

Other competence areas:

- Drawing
- Process planning
- Model shop
- Production
- Marketing
- QA
- Quality control
- Assembly
- Documentation
- Management

Some complexity aspects

- Many actors involved (> 10) in software design and implementation
- Many actors involved (> 25) in the software testing - several different perspectives
- Many differentiated and interacting components - 200000 lines code (25 applications)
- All software components are interrelated
- All design activities are conducted concurrently
- Requirements not well specified
- Hardware design and requirements change during the process
- The chemists were not certain about the algorithms design
- Lack of standards, methods
- New platform (Windows)
- Unfamiliar with the need for coordinating activities
 - support for this (classification categories, procedures, artifacts) must be invented

How should this be systematically specified?

Means for handling the coordination

- All designers placed in the same room together.
The software designers are placed next to the hardware designers
- Work plans (integrated with other plans)
- Distributed responsibilities reflecting the software architecture
- Meetings
 - ad-hoc/informal, integration, diagnosing, planning
- Platform periods (Working cycles)
 - 4 week work - 1 week integration, linking routines, procedures
- Bug reports + procedures
- Project scheduling spread-sheet

Physical properties

Procedures

Dependencies

.....

Example 2: Platform periods

Original idea:

A specific point in time where everybody "integrate their pieces"

- Standardized procedures
- Fixed roles:
 - testers, designers, spec-team, platform-master, bug report-manager
- Support tools:
 - directory structure, linking applications, etc.

Parallel processes and work-flows

Procedures

Relationships

Roles/actors

Dependencies

.....

An example: The project planning-sheet

Example 3: The bug reports

Initials:	Instrument:	Report no:
Date:		
Description:		
Classification: 1) Catastrophic 2) Essential 3) Cosmetic		
Involved modules:		
Responsible designer:		Estimated time:
Date of change:	Time spend:	Tested date:
<input type="checkbox"/> Periodic error - presumed corrected		
Accepted by:	Date:	
To be: 1) Rejected 2) Postponed 3) Accepted		
Software classification (1-5): ____		
Platform:		
Description of corrections:		
Modified applications:		
Modified files:		

Procedure:

- reporting - classification
- send to the spec-team
- diagnose - classification
- estimation of correction time
- incorporation in plans
- incorporation in bug lists and statistics
- copy -> central file
- original -> responsible designer
- correction / redesign
- original -> central file
- copy -> platform master

This is a simplification -
negotiated and refined constantly

Physical properties

(Parallel) processes and work-flows

Procedures

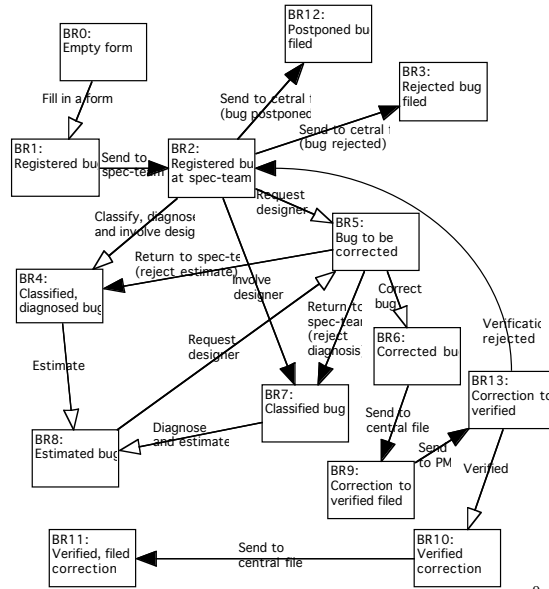
Entities and relationships

Dependencies

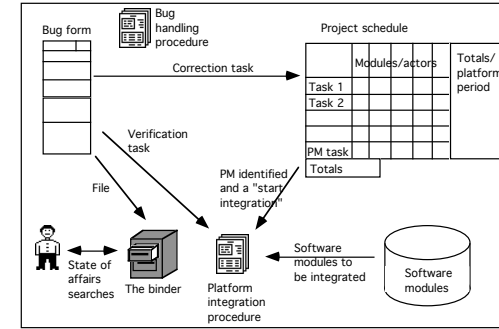
.....

An attempt:
flow and state-
specification

- States
- Transitions
- Flows
- Roles
- Parallel actions
-

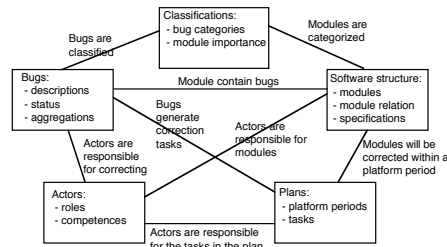


Relationships and flow at a high level



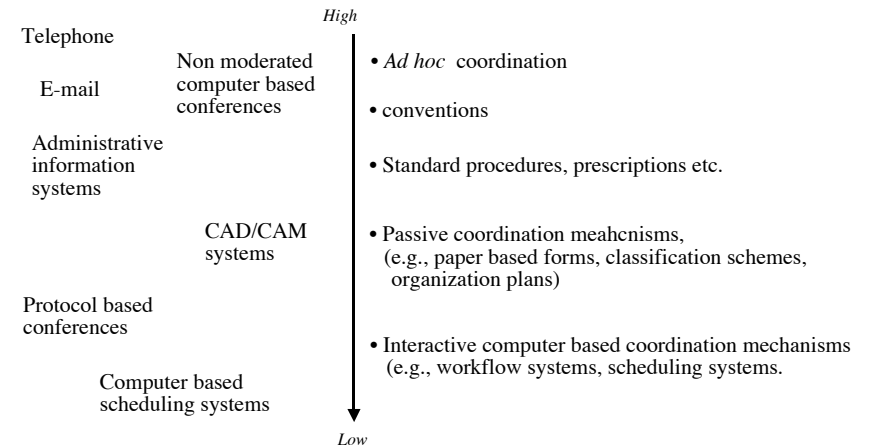
An attempt:
Entity and relationship specification

- Entities
- Relations
- Roles
-



(Carstensen, 1995)

Degree of local control



How should this be specified?

Summary

- Software development and its coordination is highly complex
- The settings and situations we must address are complex
- Modeling the settings and situations properly requires many different languages and tools
- Modeling the data and processes requires rich, but rigid languages
- Transformation of models requires "visibility"
- Physical properties
- (Parallel) processes and work-flows
- Control flows
- States and transitions
- Procedures
- Roles and actors
- Entities
- Relationships
- Dependencies
- Triggers
-