MECHATRONIC SOLUTIONS

SIOUX
SOURCE OF YOUR TECHNOLOGY

ELECTRONIC SYSTEMS

MATHWARE

TECHNICAL SOFTWARE
Informal (Behavior) Modeling for a (Medical) System

Reality can be so complex that equally valid observations from differing perspectives can appear to be contradictory.

Four

No Three

Author: Klemens Schindler
Introduction

- Context
- Anonymized example case
- Problems during project
- Gradually adding formalization (evolving tools and mindsets)
  - Formalized text and requirements
  - Graphical notation for behavior specification
  - Process integration, traceability, and status tracking
Context

- Sorry, details confidential 😞
- Medical cleaning system (V-model, medical standards such as IEC62304)
- Medium sized
  - Product family with international release (variability+regulations)
  - ±100 sensors/actuators
  - ± 300 software components (few thousand methods)
  - ± 300 error types
Example case: Parts
Example case: Program

Program:
- Rinse (hot)
- Wash
- Rinse (cold)
Problem – part 1

- Conflicting & changing statements
- Specs too high level
  - The system should clean
  - The system should have a unique selling point
- Ambiguous terminology
  - Clean: wash? rinse? spray?
  - Not qualified with critical parameters
- Focus on individual component design instead of system behavior
Problem – part 2 (feedback of domain experts)

When the spec is correct

- Sounds good, (probably)
- Well yes, OBVIOUSLY like that

When the spec is wrong

- Sounds good, (probably)
- DEFINITELY NOT like that
Text Formalization – part 1
(glossary in word processor)

Glossary

load:
Dishes in the dishwasher to be cleaned.

cleaning program:
A program designed to clean a load.
detergent:
water:
Move liquid at high speeds along the load to provide
water:
Water provided by the public water supply system.

- Duplicate terms
- Not sorted
- ...
- Maintenance hell
Text Formalization– part 2 (interaction)

- Formalize a document (e.g. glossary) in modeling environment (MPS)
- Look&Feel: IDE with “Code completion” and “checks as you type”
Text Formalization—part 3 (some features)

- **Glossary DishWasher**
  - **load**:
    - Dishes in the dishwasher to be cleaned.
  - **cleaning program**:
    - A program designed to clean a **load**.
  - **detergent**:
  - **spray**:
    - Move liquid at high speeds along the load to provide "mechanical cleaning" and bring the liquid (**water** with or without **detergent**) in contact with the load.
  - **water**:
    - Water provided by the public water supply system.

- **Error: Duplicate glossary term 'water'**
- **Warning: Consider using defined term 'load'**
- **Error: No description for glossary term**

- Not globally unique
- Not formalized
- Not described
Text Formalization – part 4

- We defined a vocabulary
- Any text we write (and review), uses that vocabulary
- A “compiler” to your document
- Details are “a hyperlink away”
- Use engineering practices on text:
  - change impact (usage) analysis
  - refactoring
Custom Formalization –
(example: sensors/actuators)

- Required fields
- Additional specific checks
  (e.g. naming conventions)
Custom Formalization – (example: actions with parameters)

Introduce a “verb/action” with “parameters” in a glossary

Informal parameters: “mention” of parameter is enough

Informal, but complete
Example case: “functions”
Graphical behavior modeling

- Look&Feel: Flowcharts (acceptable to domain experts)
- Flowchart extensions
  - reuse & parameters
  - nested contexts (background tasks/parallelism, “for each”, “controlled loops”)  
  - errors & recovery recipes
- More formal expressions instead of text found in classical flowcharts  
  (type-checked, units, decision tables, math, timed)
- Trick your experts into “almost programming” (but it doesn’t have to compile)
Graphical behavior modeling – example

- **parameters**
- **decisions with expressions**
- **expression with units**
- **parallelism (background tasks)**
- **error reporting & recovery recipe**

Diagram:
- **decide**
  - filled volume < 3 l
  - [otherwise]
- **Continuously**
  - operate spray pump
  - heat
- **InsufficientWater**
  - severity
  - error recovery
  - automatic abort
  - drain
  - rinse
  - clear liquids
- **put detergent**
  - amount
- **Wait for time:**
  - spray time
What to model

Explicitly model crucial things for “realistic/implementable” behavior such as:

- HAL (concrete sensors/actuators)
- Core data structures for process
- Critical “detail functions” (e.g. measurements)
- The most important decision tables
- Conditions and time
- Whatever is crucial for your specific case
Status tracking

- “Signatures”
  - Defined/Proposed
  - Agreed
  - Implemented
  - Tested
- Dependencies (+impact)
- Unresolved issues
- Revisions
Conclusion

- Informal models help much with concreteness, consistency and nuance
- Not a one-size fits all (adapt until you get useful feedback)
- Integration into the existing is key (mindsets, processes and infrastructure)
- Making tools is not a multi-month investment
  - Reap benefits during development, even if you intend to throw away the tool
- Even though the context is different, standard practices from software development and documentation writing apply
Further resources

- Modeling environment: JetBrains MPS
- Example of glossaries and formalized text:
  - [https://github.com/DSLFoundry/mps-rich-text-glossaries](https://github.com/DSLFoundry/mps-rich-text-glossaries)
- Contact me on linkedin
Source of your technology