Improving Software Engineering with Ontologies

modom.io in the Field

by Moritz Eberl
Our Client

- CAD Solutions for Engineers
- Tools for Planning & Documenting
- Online Parts Platform
- 35,000 Customers
Available Technologies

- Electrical Engineering
- Cabinet Engineering
- Electrical Installation
- Piping & Instrumentation
- Fluid Engineering
- Building Automation
That's a lot of Domains

• Many different Standards

• Numerous Vendors with different Solutions

• Knowledge spread through the Company
Challenges

- Developers are specializing on certain domains

Risks:
- Decreased Agility
- Knowledge Loss
Challenges

- Developers are specializing on certain domains
- Coordination Overhead

Risks:

- Uneccessary Meetings
- Slower Development
Challenges

- Developers are specializing on certain domains
- Coordination Overhead
- Existing Application

Risks:

- Implicit Knowledge
- Feature Duplication
Goals:

How to meet these challenges?

- Consolidated Data Modeling
- Defined Modeling Processes
- Documentation
- Internationalization
Current Process

Product Manager

Requirements Engineers

Feature Concept

User Story

Domain Knowledge

Suggestions

Technical Knowledge

Sales & Consultants

Software Engineers

Agile Data Modeling?
Possible Solutions

Conventional

- UML Modeling
- Wiki Documentation
- Manual adaptation to code

Problems:
- No single source of truth
- High Maintenence
- Internationalization problematic
- Provenance Data
Possible Solutions

Semantic

• Create Model in an Ontology
• Protégé as Tool
• Versioning through Git
• Manual adaptation to code

Problems:
• Limited modeling experience
• Adaptation into Code still requires manual effort
Our Solution

Modom.io
Web-Based Ontology Modeling Platform
Features

- **Simplified Modeling**
  - Describe concepts semantically
  - Reduced complexity and similar to object-orientation
Features

Simplified Modeling
Time Saving

• Load descriptions and images from DBPedia
• Translation service included
Features

- Simplified Modeling
- Time Saving
- Generation of Artifacts
  - Documentation
  - Source Code
Features

- Github
- Azure DevOps
- Simplified Modeling
- Time Saving
- Generation of Artifacts
Resulting Process

Product Manager

Requirements Engineers

Software Engineers

Domain Knowledge

Feature Concept

Suggestions

Sales & Consultants

Domain Knowledge

User Story

Domain Model

Technical Knowledge

Feature

Product Manager

Requirements Engineers

Software Engineers

Domain Knowledge

Feature Concept

Suggestions

Sales & Consultants

Domain Knowledge

User Story

Domain Model

Technical Knowledge

Feature
Resulting Process

Modom.io

Model Repository

IDE

Application Repository

Documentation (e.g. Markdown Files)

Artifact (e.g. NuGet Package)

Build

Dependency

Change

Build

Change

Build

Application
Conclusion

• Generated Artifacts were key for fast implementation
  • Code delivered a foundation
  • Documentation the right understanding

• Solid Data Migration Strategy still necessary
  • At least now changes are transparent

• Fewer Regressions
  • Testing can be directed
  • Conflicts are identified earlier
Outlook

- Loading & Linking of existing Concepts
  - AutomationML
  - ifcOWL
  - ...

- Generate all the boring code
  - Validations

- Improve usability
  - Visual Editor
Thank you!
Visit us at our booth!