
Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis

Dipl. Inform. Björn Decker

bjoern.decker@iese.fraunhofer.de

Tel.: 06301-707 183

Introduction: Some words about me



- Organization:
Fraunhofer Institute for Experimental Software Engineering
 - www.iese.fraunhofer.de
- Department: Experience Management
 - www.experiencemanagement.de
- My research interests:
 - Learning Software Organizations
 - Social Software (i.p. Wikis)

Seite 2

Agile Reuse and its Problems in a Nutshell

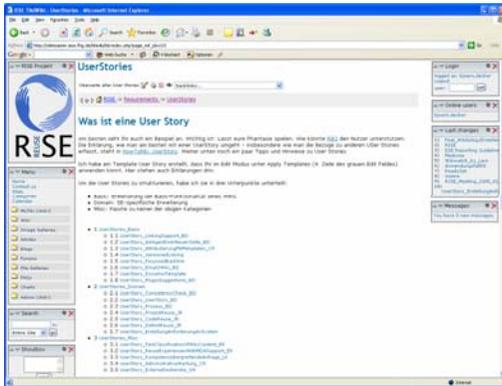


- Bottom line of reuse
 - Avoid redundant work
- Agile Reuse: Support people in reusing
 - Provide reusable artifacts when needed
 - no fixed process
 - Increasing importance of knowledge model
 - Identify people who could work together
- Problems
 - Lightweight Platform needed
 - Find: Reusable assets not known
 - Quality / consistency of entries

} Scaling

Seite 3

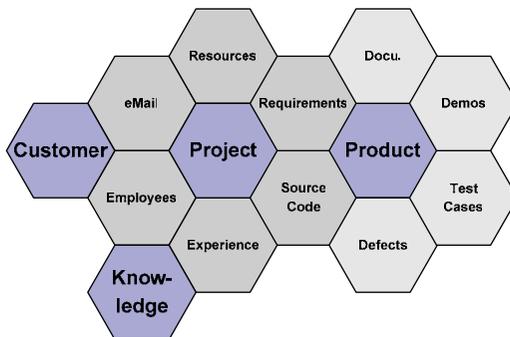
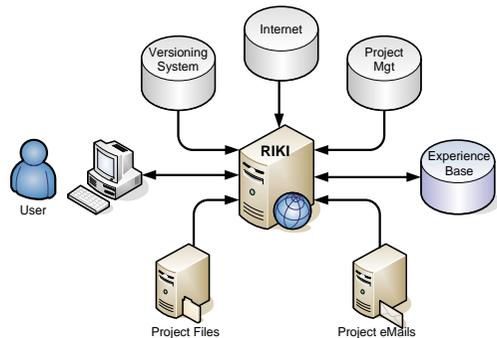
Wikis offer an agile, lightweight platform, ...



- Advantages
 - One place publishing
 - Simple and safe collaboration
 - Easy linking by page name
 - Description on demand

**... but problems “find” and “quality” remain unsolved
(second window anti-pattern)**

Solution Provided by RISE



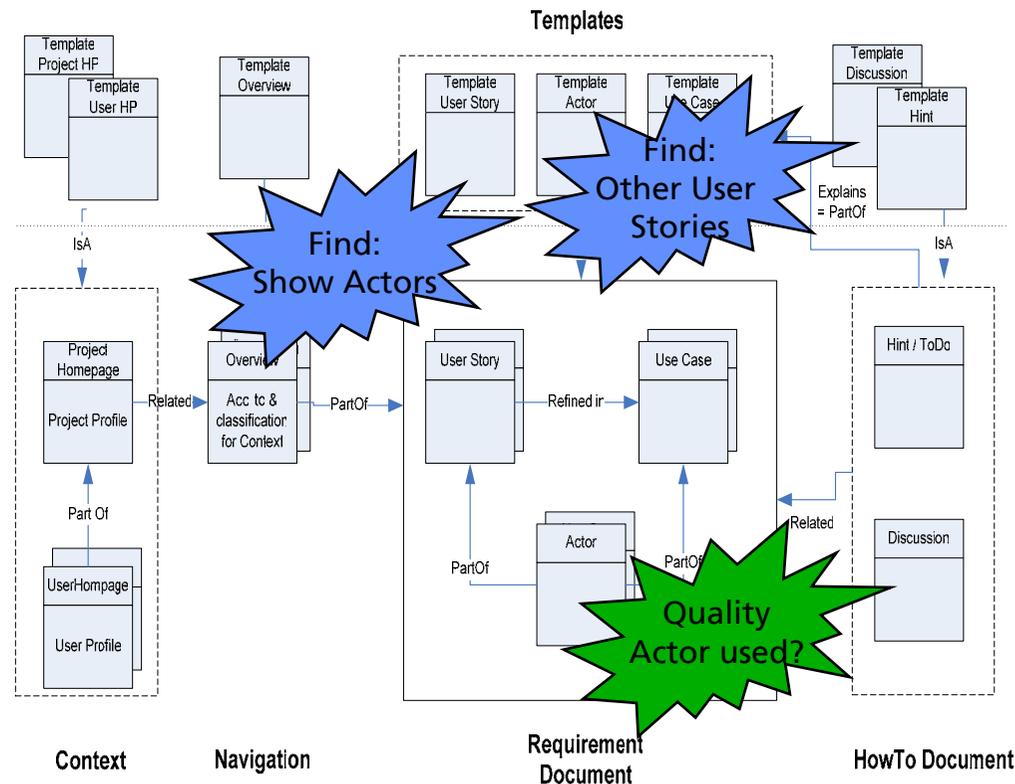
- Idea:
 - Manual / semi-automatic annotation of documents with Meta-Data (Ontologies)
 - Expected relations are defined in templates inside wiki
- Solution
 - Find: Use meta data for inference (similarity / consistency) and navigation
 - Quality: Use Meta-Data for consistency checks

Seite 5

Underlying Idea: Wikipitology

- „The Wiki is the ontology“
 - Only „vocabulary“ of ontology is defined
- The ontology of concepts is created by Wiki entries
 - Semi-automatic assignment to ontology concepts (Naming Conventions)
 - Ontology is updated by Wiki entries

Example: Requirements Engineering



• Relations

- Part-Of: "Allowed" relation (links)
- Refined in: Temporal relation
- Is-A Relation between Templates and Documents
- "Unambiguous" metadata allow "safe" reasoning
 - i.p about relations
 - Compared to Text-Search

Challenges

- Summarizing this in 10 Minutes ;-)
- Maintaining agility --> iterative learnability of ontologies
 - Finding minimal set of rules / naming conventions for the wikitology
 - Use existing structures / information
- Reuse of existing ontologies (z.B: integrate wikipedia, SWEBOK)

Current State & Promises

- Current (CBR / LSA + Tagging)
 - Show similar / relevant (also redundant) documents
 - Simple consistency checks
- Future (RDF + OWL)
 - Semantic checking of (a lot of) relations
 - Semantic traceability: From RE to source code
 - Product lines: Semi-formal modeling of variability on the requirements level → Instantiation in Wiki

Seite 9

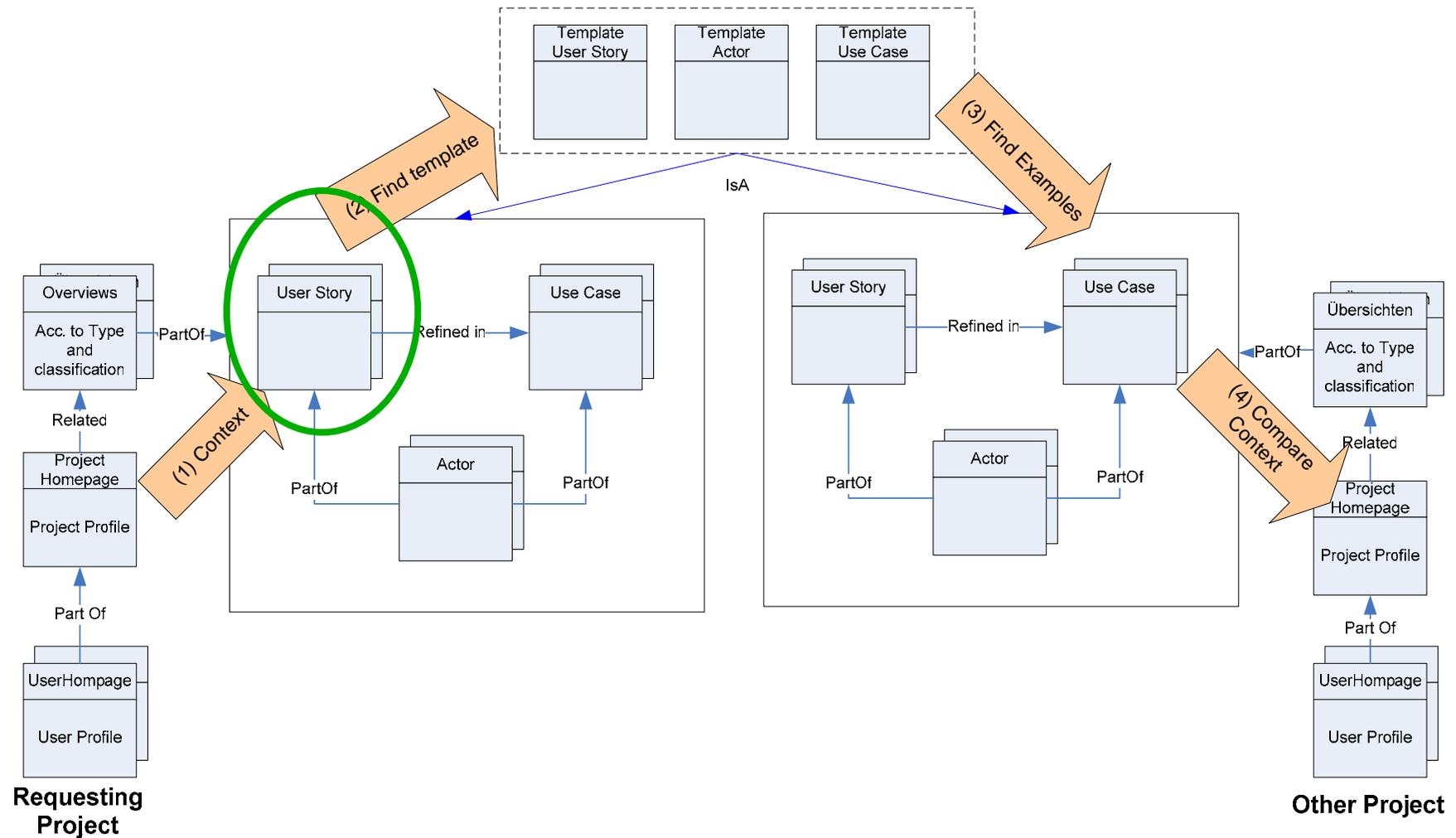
Support Folien

Example: Requirements Engineering

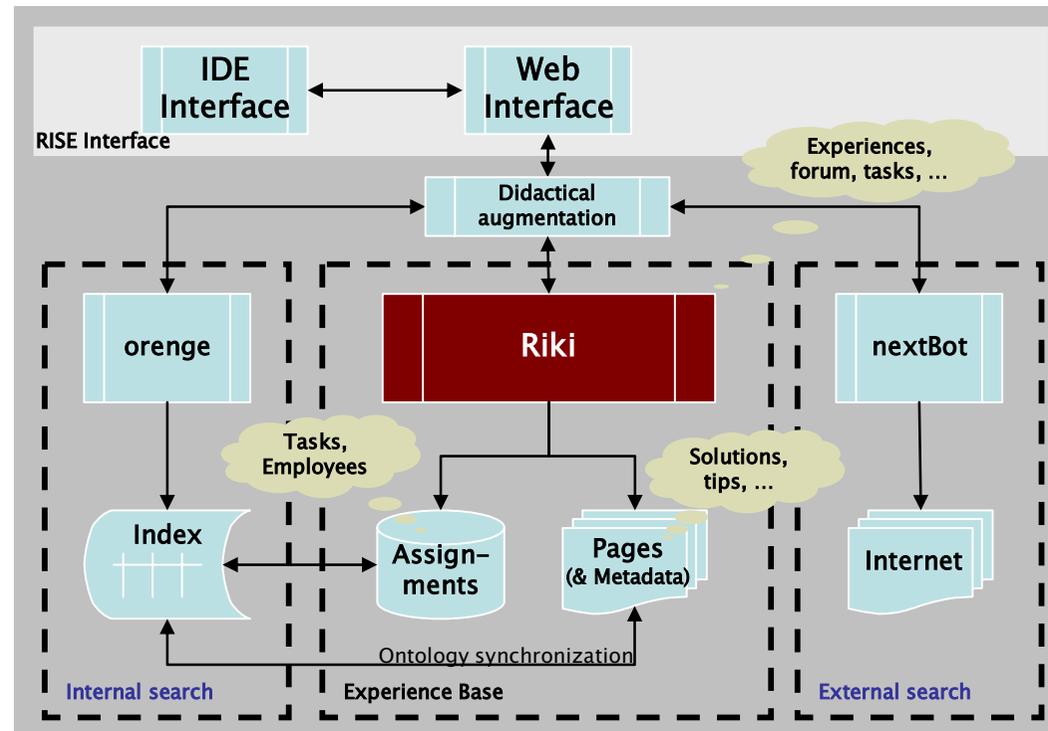
- Similar pages
 - Show Use Cases from other projects with similar project characteristics or content as current one
- Relevant Pages:
 - Show Part of relations
- Navigation
 - Show all use cases belonging to a certain Actor
- Consistency Check:
 - Is an Actor description referenced by any Use Case / User Story
- Methodology Support
 - Hints on how to create requirements
- Central Access
 - To discussions about requirements engineering
 - To code components from a certain author / referencing use case (not shown)

Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis

Example: Similar Pages



Central Access



System functionalities

- Questions the system should answer
 - How to remove defect X or test functionality Y? [Know-how]
 - Who knows the functionality in code component X? [Know-who]
 - What risks are associated with technology X? [Know-what]
 - Why was technology X used? [Know-why]
 - Where was algorithm X used? [Know-where]
 - When was technology X used? [Know-when]
- Ambient support
 - What happens if alternative technology X is used? [Know-if]
 - Which methods are concerned with problem X? [Know-that]
 - What skill do I have with programming? [special]