Multiple Views of System Specifications
connecting a distributed project

Bob Schwanke
Siemens Corporation
Corporate Research
Princeton, NJ USA
Problem

Eliciting and Managing Feature Specifications,
- Hardware and Software product
- Marketing Requirements approved
- Architecture and design in progress
- Distributed teams
  - 5 locations, trans-Atlantic
  - English- and German-speaking cultures
  - Multiple system development cultures

Hardest Parts
- Multiple audiences – by location and role
- Loose coupling to architecture
- Frequent reorganization of features

Solution: Hierarchical System Feature Model, with Tool Support
A Monitoring System

- Sensor Network
- Response Network: (Actuators and alerts)
- Automated responses
- Human-commanded responses
- Multiple markets with different drivers
- Key processor board is custom
- Second market package of product
  - Big additions to response hardware
  - Much more software than before
  - Porting to IP Network
Artifacts and Owners: a Document Dependency Diagram

Each artifact **depends on** “preceding” artifacts.

System Features
- Push back on MR’s
- NOT architecture-dependent.

Decomposed features
- VERY architecture dependent
- No push-back because same owner

Component designs
- Different owners, different locations
- MAY push back on features
Structure Among Features: Use Case Hierarchies

Feature == Use Case or Requirement

Use Case as Abstraction
- [Large] category of behavior
- Title and 1-2 sentence specification
- “The system will perform this category of behavior.”
- Control sequence unimportant

Motivation for Modeling
- Standard associations betw. features
- Gives meaning of “tree” (hierarchy)
- Breadth-first navigation for “big picture” and “wholeness”.
- Graphical presentation for meetings
- Easy to re-organize
- Modeling tool helps maintain integrity.
A Use Case Hierarchy

Two Parents
Overview of the Modeling Process

1. Elicit expert knowledge by showing the model, relaxing the rules
2. Decompose the “external” System Features into “internal” component features.
3. Refactor to align with architecture – where known.
4. Architects design iterations
5. Allocate features to iterations
6. Refactor again to reduce fragmentation
7. Tag each feature with the views in which it belongs.
8. Use tool filtering capabilities to generate views as documents.
Aligning Features with Architecture

Goal: Assign sub-features to sub-systems
- Feature decomposition **must** depend on architecture decomposition.
- Architecture not ready at beginning
- Early elicitation just captured two subclasses of responses

Architects chose
- State machine model for console
- Same responses initiated two ways

Alignment
- Refactored to separate initiation from response

Diagram:

- Respond to Condition
  - Human Controls Response
  - Fully Automatic Response
  - Analyze Sensor Event
  - Control Console State
- Execute Response

«extend» «extend» «include»
Assigning Features to Iterations
Four Views of Feature Specifications

Master View
- All Features
- Hyperlinked diagrams
- Spreadsheet Interface
- Filters
- Issues and Action Items

Marketing View
- Product Structure Overview
- Readable by non-developers
- Features that trace to MRs (Marketing Requirements)
- Clarifications of MR’s
- Rejected and Deferred MR’s
- No unnecessary detail

Architecture View
- Feature ↔ Subsystem Map
- add Engineering Requirements
- Design decisions that affect many components
- Not derived from specific external features
- e.g. communication infrastructure

Development View
- Product features
- Assigned to components
- Assigned to interfaces, iterations
- Checklist for estimation
- Complete
- No duplication or overlap ???
Views of the Model

Marketing View

Development View
Tips

1. Refine to granularity of decisions (approval, schedule)
2. Re-use MR wording where feasible
3. Mark up Marketing Requirements with clarifications
   a. You own the mark-ups, “they” still own the base document
   b. Ask Marketing to validate your mark-ups
   c. You can proceed while they revise their document.
4. Subject Matter Expert “initials” each feature.
5. SME’s can edit, validate as spreadsheet, without special tool
6. Modeling the MRs early, saves time later.
7. Plan on doing a little scripting
Summary

- Use Case Hierarchy decomposes System Features into Component-level Product Features
  - Three types of parent-child relationships
  - Child can have multiple parents
  - Attach additional specifications to use-cases
  - Example: 479 Use Cases, 267 Requirements
- Modeling Tool helps with
  - Breadth-first analysis → Check ‘wholeness’
  - Frequent re-organization of features during elicitation
  - Edit pictures in meetings with subject matter experts
  - Classify features along several dimensions
    - Iteration
    - Market Segment
    - View
  - Report generation for different audiences
Experience with Modeling Tool

Many good ideas, including
- Model-centric (not diagram-centric)
- Import/Export as Spreadsheet
- Diagram hyperlinking
- Drill-down
- Diagram filtering
- Scripting (Visual Basic)
- Custom queries in SQL
- SysML support (e.g. Components)
- Multi-user support
- Document generation
- Virtual documents
- Accidental deletion is hard
- Very rarely crashes

Still immature
- Advanced features sometimes incomplete
- Document template tools clumsy
- On-line help is weak
- Limited requirements catalog functionality
- No multi-valued attribute support
- Hard to query on links
- Can’t “round trip” to a better RE catalog