Specification and Prototyping of Building Automation Systems

Andreas Metzger

Research Group Prof. Dr. Zimmermann
Department of Computer Science
University of Kaiserslautern

Contents

System Development
Requirements Analysis
Prototyping
Environment
Case Study
Test Installation
Building Automation Systems

Characteristics:

- reactive
- complex
  - integrated control systems
- large
  - 400 – 20000 devices (sensors, actuators, controllers)
- distributed
  - geographically/logically
- many controlled effects
  - HVAC, lighting, ...

Problem:

- development of reactive, complex, large, distributed hardware and software

System Development

Process Model (V-Model):

- Requirements Analysis
- Prototype Test
- Acceptance Test
- Create Prototype
- Integration Test
- Create Useable System
- Create Environment
- Control System Software
- Operating System
- Communication System
- Control Hardware
- Design Knowledge
- Application Knowledge
- Domain Knowledge
- System Requirements
- Create System Design
- Test in Action
- Prototype
- System Design
- Environment
- Installable System
- Executable System
- Usable System
Requirements Analysis

Documents:

- problem description
  - customer's specification of system behavior (needs)
  - natural language

- system requirements
  - formalized system features
  - natural language (tasks)
  - specification language (UML, SDL, Statemate)
  - basis for contract

- application knowledge
  - building dictionary
  - domain experts

Steps:

- entry review
  - understand problem domain
  - use: application knowledge

- object structure design
  - hierarchy
  - use: building model
  - insert additional objects (e.g. controllers)

- task design
  - derive tasks from needs

- task assignment
  - assign tasks to objects
  - organizational hierarchy

- requirements structure modeling

- requirements behavior modeling
  - specify a possible realization of the functionality (needed for execution)
Requirements Structure Modeling

Generic Approach:

- apply SDL block type modeling template

Requirements Behavior Modeling

- use (extended) finite state machines
Prototyping

Prototype:

- produced with less effort
  - automatically generated from specification
- does not necessarily expose all system features
  - non-functional features can be omitted

Benefits:

- can be used for tests/validation
- enables efficient communication
- user can explore system features (can be used for req. analysis also)
- reduces overall development cost (errors can be found in early stages)

Generating the Prototype:

- ProtoCtrl
  - methods for communication with environment (via ProtoAdd)
- ProtoAdd
  - communication functions (CORBA, sockets, CAN)
- SDT Lib.
  - target platform dependent
  - integration concept (light/tight)
Prototyping

Prototype Execution:

- Environment
  - building simulator (e.g. SEMPER)
  - (partial) physical building (test installation)

- Interlink
  - different protocols
  - changing object structure and thus object names
  - dissimilar specification/programming languages
  - different handling of time
  - multiple partitions on both sides

- Results
  - feedback for design process
  - validation of customer/user requirements

Environment

Realization:

- SEMPER
  - derive DOM from building model (?)
  - derive SOMs from DOM (?)

- gebsim (SFB 501)
  - manual implementation

- PSiGene (pattern based simulator generator)
  - generation from building or object model (?)

- commercial simulators
  - not for all domains
  - missing functionality

- physical building (test installation)
Case Study

An Office Floor:

- 22 offices
- 3 hallway segments
- 578 sensors and actuators
- 360 controllers

Effort:

- total: 491h
  - structure creation: 33h
  - requirements description: 100h
  - requirements modeling: 316h
  - test: 42h

Prototyping:

- prototype generation
  - stand-alone application
  - for UNIX workstation
  - approx. 1/2 hour for whole system

- environment
  - building simulator: gebsim
  - for UNIX workstation

- interlink
  - configuration derived from list of instance-names
**Test Installation**

**Configuration:**

- **PC**
  - socket interface
  - data logging
  - web interface (manual interaction)

- **microcontrollers (8051 compatible)**
  - drive actuators
  - poll sensors
  - connected to PC through CAN bus

- **sensors**
  - temperature-, humidity-, illuminance-sensors, contacts, motion detectors
  - weatherstation (serially linked to PC)

- **actuators**
  - blinds
  - radiator valves
  - window openers
  - luminaires

**Realization:**

- 2 offices and 3 floor segments are instrumented
- outdoor light sensors on each side of building
- installation will be completed in November

**Planned Experiments:**

- replace simulator with test installation
  - prototype remains on UNIX platform

- generate prototype for PC (hardware-in-the-loop)
  - replace socket interface with CAN interface

- distribute functionality to microcontrollers
  - partitioning
  - design phase
Perspectives

Evolutionary Prototyping:
- prototype evolves into final system
- series of refinements

Distributed Prototyping:
- system is partitioned into smaller components
- single components can be validated
- earlier prototyping results

Environment:
- automatic generation from building model
- interlink with SEMPER

Summary

- efficient requirements analysis
- requirements modeling with templates
- prototyping using executable specification
- generation of environment
- test installation for further experiments