From Requirements to Architecture: Functional and Other Aspects

Eltjo Poort, LogicaCMG
(Peter de With, TU/e + LogicaCMG)
**Agenda**

- Introduction
- Models of system requirements and architecture
  - Models from literature
  - Accepted model of architectural design
  - Refined requirements classification for NFD
  - The nature of requirement conflicts
  - Three dimensions of solution strategies
- The Non-Functional Decomposition Process
- Examples
- Conclusions and discussion
Introduction
Introduction

- Primary result of software architecture process: Decomposition
  - Identifying main components
  - Relationships
  - Different views
- Question: how to derive subsystem decomposition from requirements?
  - Functional Decomposition not tailored for specific quality requirements
  - Documented methods mostly indirect (“trial and error”) or focused on specific quality attribute
- Our solution: Non-Functional Decomposition
  - Based on requirements conflicts
  - Defines trace from requirements to system structure
Motivation

• Unclarity surrounding concepts of Quality Requirements, Non-Functional Requirements
  – Which requirements determine architecture?
  – Are NFRs and Quality Requirements the same?
  – More clarity in development teams

• Disconnect between software architecture and development process
  – Hard to make trade-offs between architecture and process
  – Clash of interests between architect and project manager
Observations from experience

• Cohesive force of supplementary requirements
  – Cluster functions with similar supplementary requirements
• Divide-and-conquer conflict resolution principle
  – Separate functions that cause conflicts into different subsystems
• Entanglement of function, structure and building process
  – Three interrelated ways to fulfill requirements

• Enter: the Non-Functional Decomposition Framework
  – Combination of model and method
  – No details, points to documented solutions
  – Highlights relationships, conflicts and ways to resolve them
Models of System Requirements and Architecture
Barry Boehm (1974…): WinWin spiral negotiation model
  – architecting as a negotiation process

Yourdon (1979): Structured Design
  – functional decomposition: low coupling, high cohesion
  – architecting as a structuring process

  – quantify quality attributes, find solutions
  – architecting as a multidimensional fitting problem

Chung (2000): NFR Framework
  – architecting to satisfice “softgoals”

SEI (2000…): ADD, CBAM, QA workshops, ATAM
  – architecting as a stakeholder satisficing process
• NFRs considered leading for architectural design
Issues with accepted model

• Oversimplified relationship between quality attributes and non-functional requirements
• Ignores importance of some functional requirements in system design
• Ignores influence of NFRs on system development process
• Ignores alternatives for architecture for satisfying NFRs
• Ignores influence of implementation constraints (e.g. time, budget) on architecture
Refined Requirements Classification

- Split functional requirements into primary and secondary FRs
- Group secondary FRs with NFRs into Supplementary Requirements
The Nature of Requirement Conflicts

• Primary requirements never conflict
• Supplementary requirements and their conflicts are leading in system design

• Requirements never intrinsically conflicting, *but*:
• Solutions to one requirement often detrimental to others, e.g.:
  – Reliability ↔ affordability (light ↔ formal process)
  – Performance ↔ modifiability (low ↔ high structure)
Three dimensions of software construction

• Functional solution examples:
  – Authorization  ➔ security
  – Caching  ➔ response time

• Structural solution examples:
  – Layering  ➔ flexibility
  – Design patterns

• Process solution examples:
  – CMM practices  ➔ reliability
  – SIL practices  ➔ safety
  – CC practices  ➔ security
- **n-dimensional optimization** ➔ **3x3 solution matrix**
Benefits of refined model

- Clear relationship between requirements determining architecture versus system functionality
- Includes solution for determining development process
- Allows trade-off between development process and architecture

| Requirements to Architecture SASG | October 4, 2005 | 15 |
The Non-functional Decomposition Process
requirements to architecture SASG

Role of NFD process

- Optimize system structure for all supplementary requirements
- Iterative divide-and-conquer strategy:
  - adapt system structure to requirement conflicts
  - isolate conflicting requirements in subsystems for individual optimization
The NFD Process

• Key activities:
  – map supplementary requirements onto primary functions
  – group, split and regroup until:
    • conflicts isolated or
    • conflicts managed

• Utilize existing techniques for:
  – gathering requirements
  – fulfilling non-conflicting requirements per group
  – cost/benefit analysis of decompositions
Map supplementary requirements onto primary functions

<table>
<thead>
<tr>
<th></th>
<th>PF1</th>
<th>PF2</th>
<th>PF3</th>
<th>PF4</th>
<th>PF5</th>
<th>PF6</th>
<th>PF7</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>S2</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>S4</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>S8</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>S12</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>S13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

- In-group conflicts: conflicting requirements within group of PFs
- Grouping conflicts: PFs can be grouped in different ways
Resolving requirement conflicts

• Resolve in-group conflicts:
  – Split up functions to separate requirements
  – Repeat grouping process

• Resolve grouping conflicts:
  – Group by most important Supplementary Requirements first
  – Cost/Benefit analysis for most promising candidate decompositions
  – Supplementary Requirements that group differently become Scattered Concerns

• Any unresolved conflicts:
  – Put on risk list
  – Manage risks, e.g. outside system
  – Try aspect-oriented solutions for Scattered Concerns
Examples
Supplementary requirements (in order of priority):

SR1: Authorized access to data only (secondary function)
SR2: Reliability (quality attribute), esp. of SR1
SR3: 1-year deadline (implementation req)

SR1 applies to all data ➔

- split data from functions
- group data authorization function with data
- create subsystem “Registry Vault”

Optimize Registry Vault for reliability
Optimize other functions for implementation speed
Roadpricing System (Kilometerheffing)

• Primary functional requirements:
  – Measure position of vehicle
  – Charge based on road, time of day, distance travelled

• Supplementary requirements (in order of priority):
  SR1: Privacy: mobility patterns not deducible
  SR2: Verifiability (of correct charging by tax authority)
  SR3: Provability: enable drivers to check all data and charge

• Solution:
  – split data according to privacy sensitivity
  – keep most privacy sensitive data in vehicle for provability
  – send less privacy sensitive data to tax authority for charging
  – perform roadside spot checks for verification of correct operation
Conclusions and Discussion
Summary of NFD

• Technique to bring more clarity and structure to requirements/architecture relationship
  – Adapts system structure to requirement conflicts
  – Isolates conflicting requirements into subsystems for individual optimization

• Observations from real-world practice:
  – Helps optimize system for all supplementary requirements, including secondary functional and implementation reqs
  – Yields documented traceability between system requirements and design decisions
  – Helps communicate effects of requirements to stakeholders
  – Helps separate component responsibilities
Discussion points

• Can NFD be validated by retrospective application to successful architectures?

• Is it a feasible framework to further improve architecture process?

• Do experienced architects work this way anyway? Is it just writing down what everybody knew to begin with?

• Further work:
  – what other existing techniques is this linked to?
  – what other possible areas of application are there?