Intro to Architecture Intent and Rationale

Matthew J. Hawthorne
ENS 509B

Software Architecture Review

- Architecture = {Elements, Form, and Rationale}
  - Perry-Wolf '89, '92
- Elements
  - {Components, Connectors}
- Form
  - Structure of: {Components, Connectors}
- Rationale
  - Reasons for selecting: {Elements, Form}

Software Architecture Review

- Elements and Form
  - The "What" of the Architecture
- Rationale
  - The "Why" of the Architecture
- Focus on Elements, Form
  - Components and Connectors
  - Architecture Descriptions
    - ADLs, UML, etc.
    - IDEs
  - Implementation Domain Concerns
- Rationale
  - Implicit
  - Informal
  - Post-priori

Rationale

- Some effects of "missing" Rationale
  - Lack of traceability from Requirements to Architecture
    - Difficult to ensure a given Architectural Design fulfills a given set of Functional Intent as defined in the system Requirements
    - Impossible to reason about optimality or other qualities of Architectural Design
  - Lack of traceability from Architectural Elements to Requirements
    - Difficult to match candidate Architectural Elements to Requirements (e.g., open source or COTS components that could be incorporated into the Architecture)

Rationale-Based Architecture

- Goal: Use Rationale as the core concept directing Architectural Design
- Current approaches
  - Architectural Styles and Patterns
    - Generalized {Element, Form} schemes to handle Implementation Domain concerns (e.g., Client-Server, Layered/N-Tier, Service-Oriented Architectures (SOA), etc.)
  - Intermediate Decision-based Models and Views between Requirements and Architecture
    - Create an intermediate model between Requirements and Architecture (e.g., Grunbacher04, Jansen05)
    - Capture Architectural Design Decisions

Rationale as Basis for Architecture

Intuition:

- Requirements define Functional Intent of system
- Desired system Functionality (i.e., Functional Intent) should form the basis for system architecture
- Issues, e.g.:
  - Impedance mismatches between units of FI and AEs
  - 1:N, granularity mismatches, etc.
- Rationale forms the basis for logically valid mappings/transformations from Functional Intent to Architectural Intent, as expressed by sets of Architectural Entities (AEs, i.e., Components and Connectors)

Rationale-Based Architecture: Theoretical Basis

Transformation from Goals (G) to Arch. Elements (E)

- Foreach ({*G*}, {*E*}), *Trans* ({*G*} ► {*E*})

Form: Mapping sets of Constraints (C) and inter-Element Relations (R) to sets of Elements (E)

Foreach ({*C*}, {*R*}):

- Map ({*C*} ►{*E*})
- Map ({*R*} ►{*E*})

Given sets of (E, C, R, and Mappings, M):

- *Form* = (*E*, *C*, *R*, *M*)

Theoretical Basis

- Rationale: Logical reasoning behind mappings from a set of Goals and Constraints in the problem domain to a set of Elements and Forms in the Architectural Domain (i.e., the HL system design)
- Rationale uses
  - Sets of Goals (G), Constraints (C) & their relations (R)
  - Formal Rationale Models (Mr) & domain-specific factors (D)
- to derive/justify
  - transformations into Arch. Elements (E) and Form (F):
- *Map* $\{G_i, C_i, R_i, Mr_i, D\}$ ► $\{E_i, F\}$

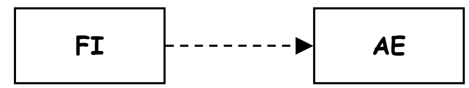
Theoretical Basis

➔ *Rationale* = (*G*, *C*, *R*, *M*, *Mr*, *D*, *Ia*)

➔ Where
- *G* = set of Goals
- *C* = set of Constraints
- R = set of Relations
- *M* = set of Mappings
- Mr = set of Rationale Models
- D = set of Domain-specific conditions
- Ia = set of Architectural Intent (reasons for Mappings)

Rationale Transformations

➔ Direct Functional Mapping (1:1)
- Map a unit of Functional Intent (FI) to a single Architectural Entity (AE)

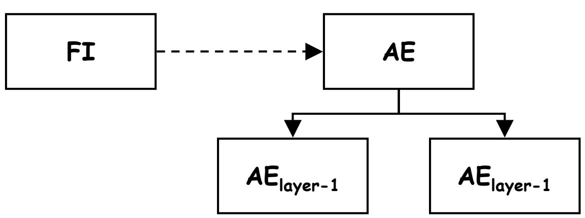


➔ Reason = Functional Intent

➔ Constraint = AE.FI = FI

Rationale Transformations

➔ Functional Decomposition (1:N)
- Decompose a unit of Functional Intent into N lower-level Architectural Entities
- "Vertical" decomposition (e.g., N-Tier architectures)

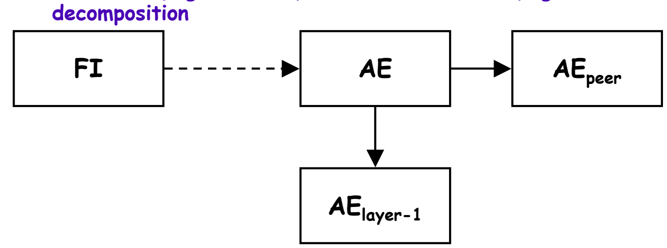


➔ Reason = Excessive Func. Intent / Diverse Concerns

➔ Constraint = AE.FI = $\sum FI_i$.FI; AE.FI = FI

Rationale Transformations

➔ Separation of Concerns (1:M+N)
- Separate a unit of Functional Intent into M peer + N lower-level Architectural Entities (M>=0, N>=0)
- "Vertical" (e.g., N-Tier) and/or "horizontal" (e.g., P2P/SOA) decomposition

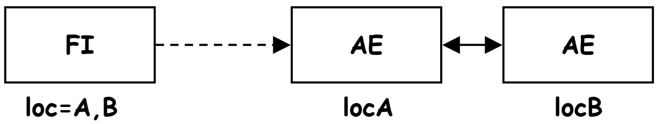


➲ Reason = Diversity of Concerns

➲ Constraint = $\sum$AE.FI = FI;

Rationale Transformations

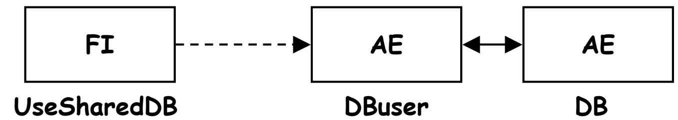
- ➔ Spatial Mapping (1:M+N)
  - Map Functional Intent based on Spatial/Geographical requirements
  - "Horizontal" division of FI
  - E.g.: Client-server, peer-to-peer, web services



- Reason = Geographic distribution of functionality
- Constraint = AE.loc=AE.FI.loc; ΣAE.FI/loc=ΣFI/loc;

Rationale Transformations

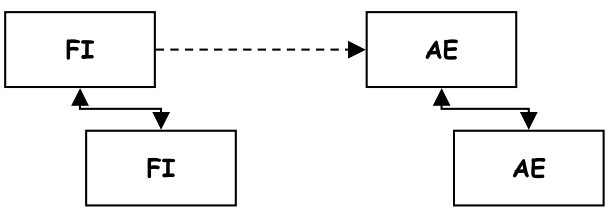
- ➔ Resource Sharing / Data-Layer Integration
  - Map Functional Intent based on shared resources (e.g., DB)
  - "Horizontal" or "vertical" division of FI
  - E.g.: Client-(DB)server, peer-to-peer, web services



- Reason = Integration +Separation of Concerns
- Constraint = AE_{DBuser}.use(AE_{DB})

Rationale Transformations

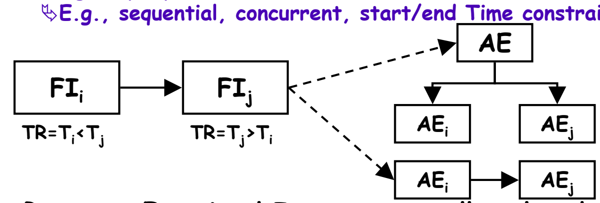
- ➔ Functional Relation Mapping
  - Map logical relations between units of Functional Intent to logically equivalent relations between Architectural Entities
  - "Horizontal" or "vertical" division of FI



- Reason = Functional Intent logically related
- Constraint = AE_i.Rel(AE_j) $\equiv$ FI_i.Rel(FI_j)

Rationale Transformations

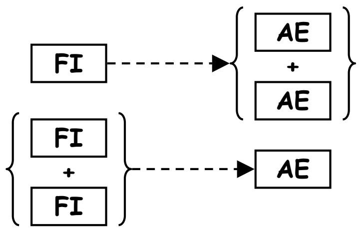
- ➔ Temporal Relation Mapping
  - Map temporal relations between units of Functional Intent to logically equivalent relations between Architectural Entities
  - E.g., sequential, concurrent, start/end Time constraints



- Reason = Functional Intent temporally related
- Logical-temporal: AE_j depends on the result of AE_i
- Concurrency: AEj depends on AEi being active (i.e., "listening")
- Constraint = AE_i.T-AE_j.T = FI_i.T-FI_j.T

Rationale Transformations

- Refactoring(1:M+N)
  - Refactor one or more units of Functional Intent to match granularity of Architectural Entity Intent
  - Decomposition or Combination



- Reason = Incorporate existing Architectural Entity
- Constraint = ΣAE.FI >= ΣFI

Mapping Functional Intent to AEs

- Current practice: Implementation object-based system design
  - AE selection/definition by implementation object instance
  - No relation to Functional Intent

- A Better Model for System Design

1. Common model of Functional Intent between Requirements and System Design
2. Rationale model to ensure that FI mapping to AEs preserves logical FI relations

Rationale Reification

- *Reification*: To *reify*, or realize an abstract system design

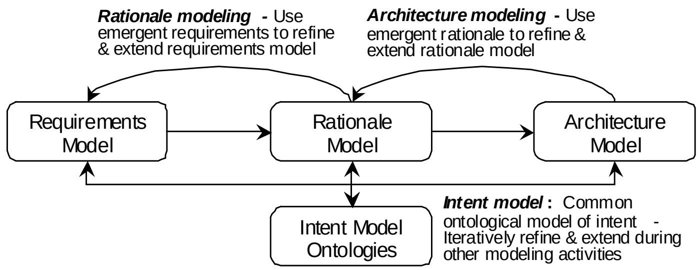
- *Rationale* Reification:
- Rationale-based modeling approach
- Abstract system design based on *Functional Intent* (instead of implementation domain objects)
- Rationale: Mappings/Transformations from Functional Intent to Architectural Entities
- System Architecture
  - Abstract
  - Rationale-driven
  - Intent-based

Rationale Reification

- Overview:

1. Common model of Functional Intent shared by both Requirements and System Design
2. Model Requirements as Functional Intent (i.e., define new FI entities or assign to existing equivalent FI if available)
3. Tag/classify all Architectural Entities and implementation objects using the same FI model
4. Use intermediate Rationale model preserving details of all FI -> AE mappings/transformations to ensure that AEs, AE relations logically preserve FI, FI relations

Rationale Reification Approach



Rationale Reification

- ➔ Iterative, model-based approach
- ➔ "Forward" modeling from Requirements (FI) to Architecture (AEs) via Rationale
- ➔ "Backwards" modeling from Architectural Entities (AEs) to Functional Intent (FI) of the Requirements
- ➔ Iteratively refine model in both directions

Advantages of Rationale Reification

- ➔ Traceability
  - Requirements (FI) → Architectural Elements (AEs)
  - Architectural Elements (AEs) → Requirements (FI)
  - Rationale
    - → Source Requirements (FI)
    - → Target Architectural Elements (AEs), Styles and Patterns
  - Requirements (FI) and Architectural Elements (AEs), Styles and Patterns → Rationale

Advantages of Rationale Reification

- ➔ Reasoning about Architecture, e.g.:
  - Properties of Architecture
  - Rationale
    - ➢ Architecture
    - ➢ Architectural Features
  - Conformance to Requirements
    - ➢ Functional Goals
    - ➢ Non-Functional Constraints

Advantages of Rationale Reification

- ➔ Reuse: Can reuse all model entities, e.g.:
  - Functional Intent (FI) → Requirements
    - Goals
    - Constraints
    - Relations

Advantages of Rationale Reification

- ➔ Reuse (cont.)
  - Rationale Elements (REs) → Design decisions
    - *Reasoning* behind architectural design decisions
    - *Context* for decisions: Links to relevant elements/properties of:
      - ✓ *Problem domain* elements & properties motivating arch design
      - ✓ *Architectural domain* elements & properties motivating arch design
    - *Result* of decisions
      - ✓ *Source* functional intent element(s), constraint(s)
      - ✓ *Target* AE(s), form, relations, styles, patterns

Advantages of Rationale Reification

- ➔ Reuse (cont.)
  - Architectural Entities (AEs) → Architectural designs
    - Elements: Components and connectors
    - Form: Structure of elements
    - Relations
    - Interactions
    - Styles & patterns

Rationale Reification Process

- ➔ Intent Model
  - Model functional intent along multiple dimensions or aspects
  - Map every requirement to one or more of the following:
    - Functional intent (FI) element
    - Functional intent (FI) relation
  - Identify logical relations between requirements/FI elements, map logical relations to:
    - Functional intent (FI) relation
    - Functional intent (FI) region
      - ✓ Extent or area of FI covered by region
      - ✓ Property or aspect of FI to which region applies
      - ✓ Constraint or relation to apply to FI property or aspect in region

Rationale Reification Process

- Rationale Modeling Process
  - Map functional intent (FI) elements to abstract architectural entities (AEs)
    - Define AEs using the same functional intent (FI)
    - Emergent architectural intent (AI)
  - Map FI relations & regions to AE relations
    - Define AE relations, regions using same FI
    - FI regions generally map to multiple AE relations
  - Derive rationale for architectural transformations from
    - Properties of functional intent (FI)
    - Relations among functional intent Rel(FI)
    - Non-functional constraints (NFC)
    - Constraint regions (R)
    - Emergent architectural intent (AI)

Rationale Model

- Rationale ID
  - Unique identifier for a given rationale element
- Rationale classification
  - Semantic classification(s) of rationale, i.e.:
    - What family(ies) of rationale this RE belongs to
    - What kind(s) of rationale this RE is
  - Problem type
  - Solution type

Rationale Model

- Transformation: Description of transformation
  - Sources
    - Functional intent (FI) element(s)
    - Functional intent (FI) relation(s) (Relation(FI))
    - Non-functional constraints (NFC)
  - Results
    - Architectural element(s) (AE(s))
    - Architectural element (AE) relations (Relation(AE))
    - Architectural element (AE) form (Form(AE))

Rationale Model

- Context
  - Problem Domain context
    - PD Elements motivating architectural design
      - ✓{FI, NFC}
    - PD Properties motivating architectural design
      - ✓{FI.Property, NFC.Property}
    - PD Relations motivating architectural design
      - ✓{Relation(FI) Region(FI), Form(FI)}
  - Architectural Domain context
    - AD Elements motivating architectural design
      - ✓{AE}
    - AD Properties motivating architectural design
      - ✓{AE.Property}
    - AD Relations motivating architectural design
      - ✓{Relation(AE), Form(AE)}