Transforming Requirements Specifications into Architectural Prescriptions

by

Manuel Brandozzi
and

Dewayne E. Perry
Introduction

- Requirements to Architecture traditionally the most difficult step in Software development
- Need for a formal approach
- We propose the use of architectural prescriptions to make the step easier
Outline

- Introduction to Architectural Prescriptions
- Overview of KAOS
- Overview of APL, the Architectural Prescription Language
- Our current work on transforming a KAOS specification into an APL specification
What is an Architectural prescription?

- It’s the architectural design of a system in terms of:
  - The constraints on the system’s components (expressed in the vocabulary of the requirements)
  - The interrelationships among these components

- An Architectural Prescription provides the basic framework to satisfy the requirements
What is an Architectural prescription? – cont.

- An Architectural Prescription can be seen as the least constraining Architectural Description
- A two step architectural process
  - Requirements to APL
  - APL to a more complete architectural description
Requirements specification for an airport:

- Goal: manage incoming and outgoing passengers
- Sub-goals: control flights, welcome passengers, manage luggage, …
Architecture Prescription

Architectural Prescription for an airport:

- Components: flights controller, passenger facilities, luggage manager, …
- Constraints on flight controller: monitor incoming and outgoing flights, coordinate them, …
- Constraints on passenger facilities: ...
- …
Architecture Description

- Architectural Description for an airport:
  - Components: control tower, waiting rooms, ...
  - Constraints on control towers: the constraints on flight controller plus the specifications for a control tower
  - Constraints on waiting rooms: ...
  - ...

Differences between a Prescription and a Description

- A Prescription makes the step from requirements to architecture easier to perform and to formalize
- Problem domain vs. Solution domain
- It enables the use of new, innovative solutions
- It enables a higher degree of reusability by fewer constraints on the solution space
KAOS

- A goal oriented requirements specification language
- The domain is modeled as objects and operations
- First the goals of the overall system are specified
- Then these goals are refined till their sub-goals are achievable by some agents
Component (name of component):

KAOS spec.: (name of specification)

Type: [Software system, Processing, Data, Connector]

Constraints: (constraint,)*

Composed of: (sub-component,)*

Uses: (component,)*, ((connector component)* to interact with {(component)})*
From KAOS Specs to APL Specs

- **Agent**
- **Entity**
- **Relationship**
- **Goal**

- **Process / Connector**
- **Data component**
- **Data / Relationship among components**
- **Constraint on the system or on a subset of the system / one or more additional architecture components**
From KAOS Specs to APL Specs – cont.
Future Work

- Formalize techniques to take into account non-functional requirements (reliability, performance, reusability, etc.)
- Extend our approach to other kinds of formal requirements specifications
- Perform experiments to confirm and elaborate our hypotheses
**Component**: SchedulerManager:

*KAOS spec.*: MeetingPlanner

*Type*: Processing

*Constraints*: Achieve[SchedulerAvailable]

*Composed of*: /

*Uses*: Scheduler,

Mconnector *to interact with* {Scheduler}