On ADLs and tool support for documenting view-based architectural descriptions

Danny Weyns
Alexander Helleboogh

SATURN 2008, Software Engineering Institute, CMU

DistriNet Labs

• @ Dept. Computer Science K.U. Leuven
• +60 researchers
• Advanced open and distributed software
• Decentralized software architectures
Outline

- Case
- Problems with view-based architectural documentation
- First step towards a solution
- Reflection & conclusions

Case: Traffic Control System

Detectors  Control System  Actuators
View-Based Documentation of Architecture

Top Level Module Decomposition View

Traffic Management: Module Decomposition View
View-Based Documentation of Architecture

Mapping between views

<table>
<thead>
<tr>
<th>System Module Decomposition View Element</th>
<th>Traffic Management Module Decomposition View Element</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detector Management</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Data Management</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Traffic Management</td>
<td>- Traffic Model - Incident Detection - Solutions Strategies - Traffic Information</td>
<td></td>
</tr>
<tr>
<td>Actuator Management</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

The elements from the System Module Decomposition View contain the modules from the Traffic Management Decomposition View.

View-Based Documentation of Architecture

Traffic Information: Context Diagram

Solution Strategies

Traffic Model

Model Info

Notification

Traffic Info

Incident Decision

Incident Info

Traffic Info

User

NOTATION

Entity under consideration

External entity

Data flows from X to Y

Diagram notation and symbols:
- Solution Strategies
- Traffic Model
- Model Info
- Notification
- Traffic Info
- Incident Decision
- Incident Info
- User
Outline

- Case
- Problems with view-based architectural documentation
- First step towards a solution
- Reflection & conclusions
Update View-Based Documentation

Add module: Media Proxy

Applying Views & Beyond can be hard

- Maintaining consistency among
  - Views
  - Mapping between views
  - Related view packets (parent/child; sibling)
  - Context diagrams
  - Overlays
  - Elements directory
  - Glossary
  - ...
- But also analysis & verification, traceability, etc.
The Underlying Problem

Common agreement: software architecture should be documented with multiple views

Lack of ADLs and Tools with explicit support for views & view relations

Our experience: documenting view-based descriptions is hard

Gap

Outline

• Case
• Problems with view-based architectural documentation
• First step towards a solution
• Reflection & conclusions
First Step Towards a Solution

- Extending existing ADL/Tool with support for
  - Views
  - View relations
- Current focus
  - xADL and ArchStudio
  - Structural views + basic relations
  - Formal foundation
- Cases
  - Prototype applications
  - Projects with industry
    - Logistics
    - Traffic control (Flexsys project)

Demo

ArchStudio 4
Outline

• Case
• Problems with view-based architectural documentation
• First step towards a solution
• Reflection & conclusions

Reflection

• Large body of work on views & view relations
  → Need for promoting views & view relations as first-class citizens in ADLs
• ADLs & Tools amplify the power of views & view relations
  → Automated consistency checking
  → On-the-fly composition of views
• Supporting views & view relations is complex
  → Relating views ≈ relating stakeholder perspectives
  → Heterogeneity of multi-view specifications
  → Formalizing the specification
Conclusion

Bridging the gap


Architectural Blueprints—The “4+1” View
Model of Software Architecture
Philippe Kruchten
Rational Software Corp.

Abstract
This article presents a model for describing the architecture of software-intensive systems, based on the use of multiple, concurrent views. This use of multiple views allows to address separately the concerns of the various ‘stakeholders’ of the architecture: end-user, developers, systems engineers, project managers, etc., and to handle separately the functional and non functional requirements. Each of the five views is described, together with a notation to capture it. The views are designed using an architecture-centered, scenario-driven, iterative development process.

Keywords
software architecture, view, object-oriented design, software development process

Introduction
We all have seen many books and articles where one diagram attempts to capture the gist of the architecture of a system. But looking carefully at the set of boxes and arrows shown on these diagrams, it becomes clear that their authors have struggled hard to represent more on one blueprint than it can actually express. Are the boxes representing running programs? Or chunks of source code? Or physical computers? Or merely logical groupings of functionality? Are the arrows representing compilation dependencies? Or control flows? Or data flows? Usually it is a bit of everything. Does an architecture need a single architectural style? Sometimes the architecture of the software suffers scars from a system design that went too far into prematurely partitioning the software, or from an over-emphasis on one aspect of software development: data engineering, or run-time efficiency, or development strategy and team organization. Often also the architecture does not address the concerns of all its “customers” (or “stakeholders” as they are called at USC). As a remedy, we propose to organize the description of a software architecture using several concurrent views, each one addressing one specific set of concerns.

An Architectural Model
Software architecture deals with the design and implementation of the high-level structure of the software. It is the result of assembling a certain number of architectural elements in some well-chosen forms to satisfy the major functionality and performance requirements of the system, as well as some other, non-functional requirements such as reliability, scalability, portability, and availability. Perry and Wolfe put it very nicely in this formula, modified by Boehm:

Software architecture = {Elements, Forms, Rationale/Constraints}

Software architecture deals with abstraction, with decomposition and composition, with style and esthetics. To describe a software architecture, we use a model composed of multiple views or perspectives. In order to eventually address large and challenging architectures, the model we propose is made up of five main views (cf. fig. 1):

• The logical view, which is the object model of the design (when an object-oriented design method is used),
• the process view, which captures the concurrency and synchronization aspects of the design,
• the physical view, which describes the mapping(s) of the software onto the hardware and reflects its distributed aspect,

4+1