ACME

A powerful ADL

Project 2
OP5

ADL presentation

Euro Team

Alauzet Pierre, Ahvenniemi Mikko,

Colin Julien, Starck Benoit

KAIST

TABLE OF CONTENTS

1. WHY DO WE USE ADLS ?

- Necessity of using standardized architectural representation
  - ADLs bring standards for architecture description, just as what
    - UML do for design
    - Entity-relationship model do for database
  - Using architectural styles for the structure
    - Pipe and filters
    - Client/Server
    - …
  - Using formal language
    - Components
    - Connectors
    - …
  - Makes the architecture universally understandable
    - Designers
    - Programmers
    - Stakeholders

1. WHY DO WE USE ADLS ?(CONT.)

- An ADL is a language for modeling a software system's conceptual architecture, distinguished from the system's implementation

- ADLs bring the tools for architecture evolution and reusability

- Makes the architecture assessable using external tools or methods

2. ADL INVESTIGATION AND CHOICE

- Comparison between several available ADLs
  - Abacus :
    - Not well provided (movies, tutorials,…) but not a lot of papers
    - For professional & enterprises
    - Software : 30 day trial
  - Rapide :
    - Best documentation and example
    - Software on Linux & Solaris : free BUT NOT ACCESSIBLE
  - Wright
    - Not enough documents
    - Well represented on Internet
    - No software
  - Unicon
    - Good specification
    - Software not available

2. ADL INVESTIGATION AND CHOICE (CONT.)

We have chosen ACME !

- Well known
- A lot of documentation
  - Website
  - HTML large documentation
  - Tutorials available
- Complete and well-made software as an Eclipse plug-in
- Free software available on every platforms
- Developed in the same university as ATAM method – *Carneggie Melon University*

3. WHAT IS ACME ?

- Acme created in 1995 by Carnegie Mellon University

- The original goal was to provide a **common language** that could be used to support the interchange of architectural descriptions between a variety of architectural design tools.

- Provide a generic, extensible infrastructure for describing, representing, generating, and analyzing software architecture language description.

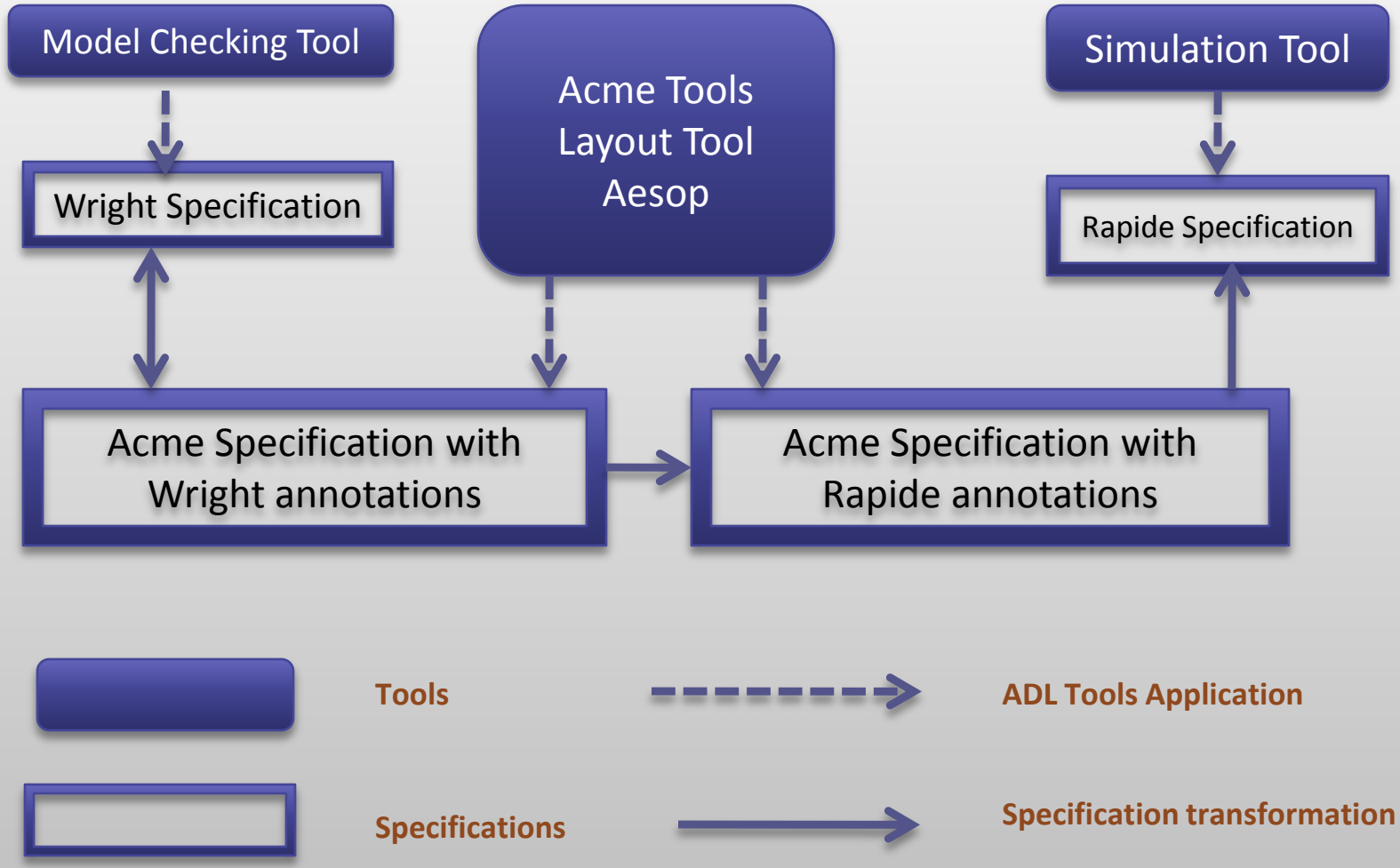
- Provide descriptions that are easy to understand for everyone

3. WHAT IS ACME ? (CONT.)

- One of ACME's goals is to be an ADL interchange format
  - Facilities exist for translating ACME to Aesop, Rapide, Wright, and back

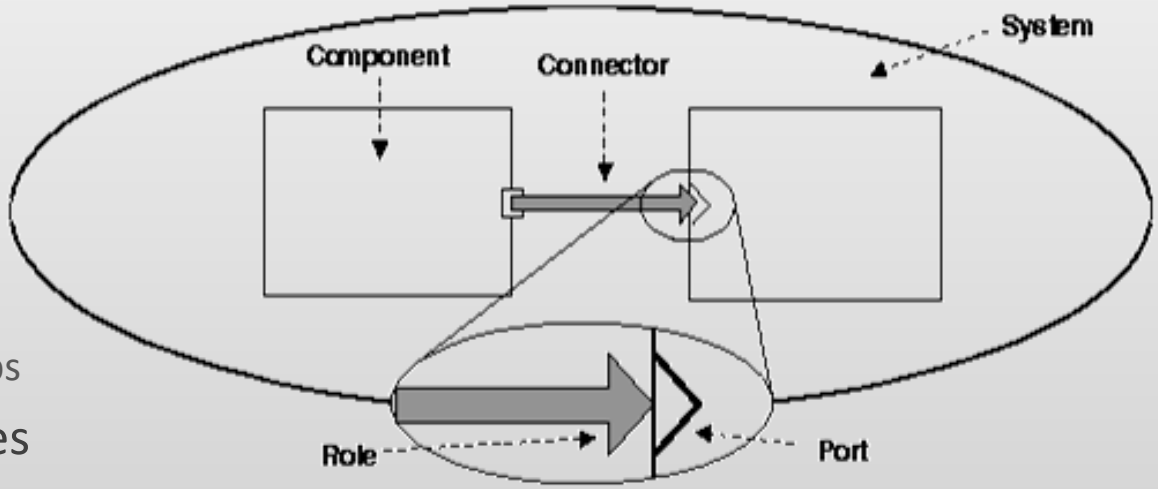
- Some steps have to be taken to take full advantage of this (e.g. Wright to Rapide)
  1. Translating specification form ADL 1 to ACME specification
  2. Translating the annotated ACME specification in ACME specification with ADL 2 annotations
  3. Finally, ACME code can be directly translated into ADL 2 specific description

3. ACME TRANSFORMATION



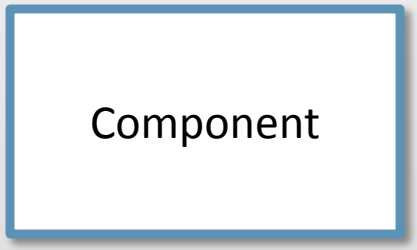
3. ACME DESCRIPTION

- ACME describes a whole system thanks to
  - Library of 7 architectural elements
    - Components
    - Connectors
    - Systems
    - Ports
    - Roles
    - Representations
    - Representation maps
  - Architectural families
    - Tiered
    - Pipe & filters
    - Client & servers
    - Pub-Sub
    - Shared data
    - Three-tiered



3. ACME DESCRIPTION

- Component
  - Primary computational elements & data stores
    - Filter
    - Object
    - Client/Server
    - Database
    - Black board

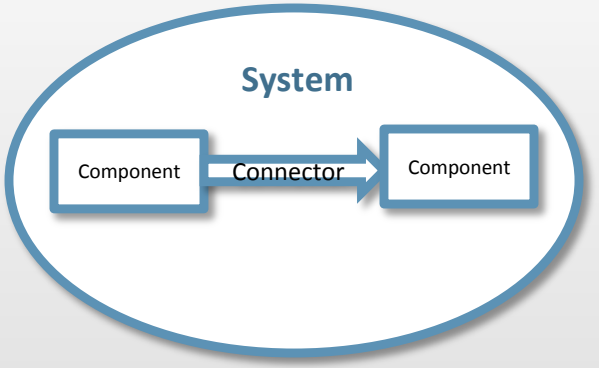


- Connectors
  - Interaction among components
    - connector embodying HTTP protocol within a client/server architecture
    - data flow channel in a pipe/filter architecture

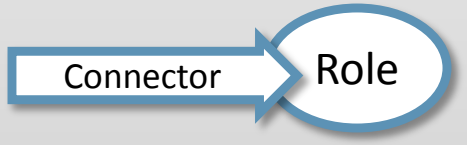
  - Communication & coordination among components
    - asynchronous communication channel such as event bus

3. ACME DESCRIPTION (CONT.)

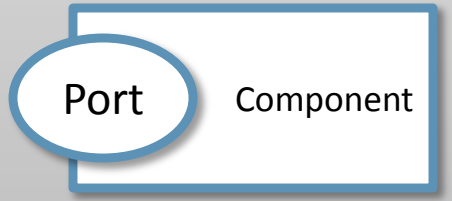
- Systems
  - First ordered entity in ACME
  - Configuration of components & connectors



- Role
  - Particularity of the connector
  - Describes how the connector links the components
    - Client-server connector has 2 roles designated caller & callee
    - Reading or writing pipe



- Port
  - Anchorage point on the component
  - Describes input or ouptut of a component
  - Can be unique or multiple



3. ACME DESCRIPTION (CONT.)

- Properties
  - Used to define component's or connector's behavior
  - Provide a way to encoding information to be interpreted
  - Defines information that are likely to change within the architecture
  - Properties are transparent for ACME itself

- Examples of ACME code and properties definitions
  - Example of code describing a filter architecture and his behavior defined as a propriety to be read by a Java IDE for implementation.

```
Component TheFilter = {
Port in;    Port out;
Property implementation :
String ="while (!in.eof) {
in.read; in.read; compute; out.write}";}
```

3. ACME DESCRIPTION (CONT.)

- Example of server component. The defined property represents a non-functional requirement. We want the server to respond in less than 15 ms.

```
Component Server = {
Port requests;
Property responsetime :
Float = 15.00 << units="ms">>;}
```

- This last example shows us how to rely an architectural item in ACME to another ADL

```
Component TheFilter = {
Property external-type :
"SomeADL::Filter";
Port in;   Port out;;}
```

3. ACME DESCRIPTION (CONT.)

- Types
  - A type is a ready-to-use structure prototype that can be use as an architectural template
  - The architect is allowed to create his own types
  - Types includes information that is not likely to change within the architecture

- Example of type definition

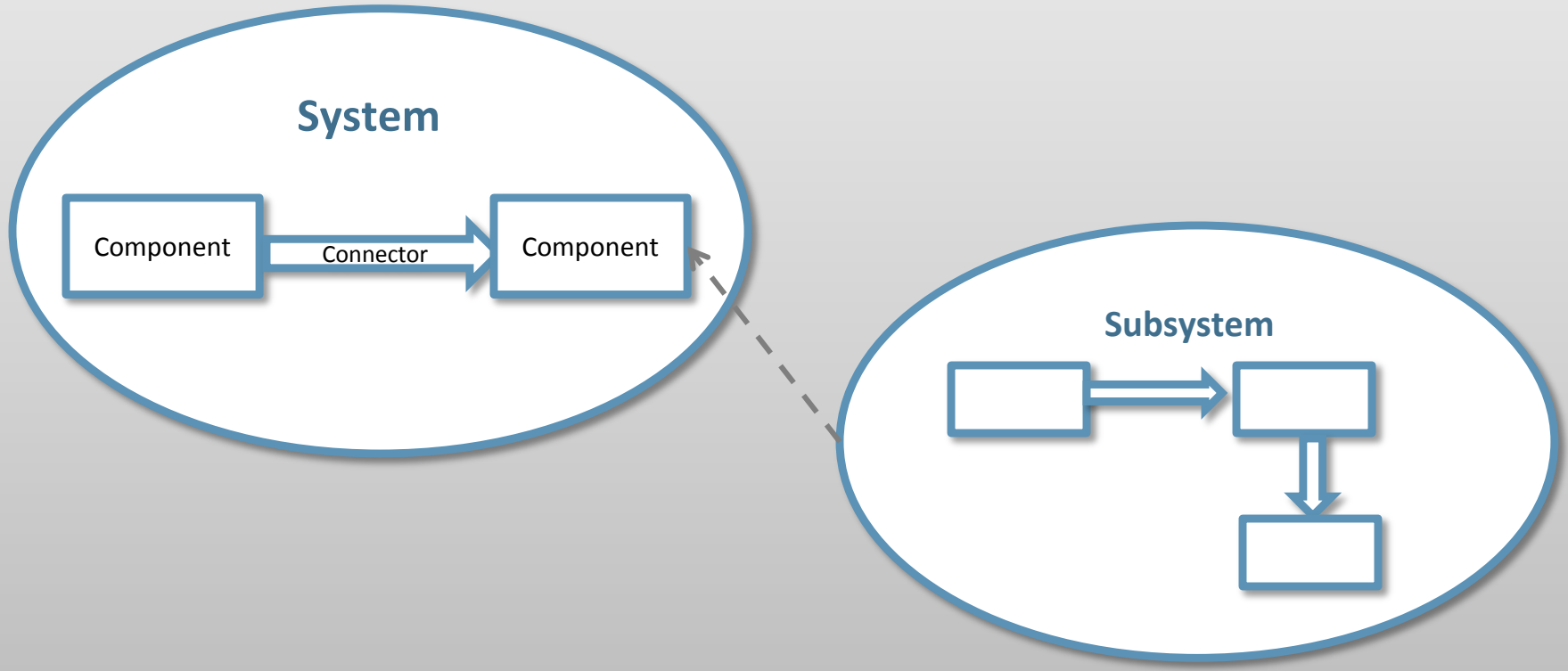
```
Component Type EventListenerT = {
Property eventMap; Property
implementation; };
```

3. ACME DESCRIPTION (CONT.)

- Language is based on
  - First order predicate logic
  - Rules checking if architectural model is well formed
- Rules can be defined
  - By the style designer
  - By ourself
- 2 types of rules
  - Invariant : violations of which are errors
  - Heuristics : violations of which leads to warnings

4. ACME PARTICULARITIES

- Representation
  - Way to abstract complex system
  - Lower level view of a component
  - Component contains & represents a sub system

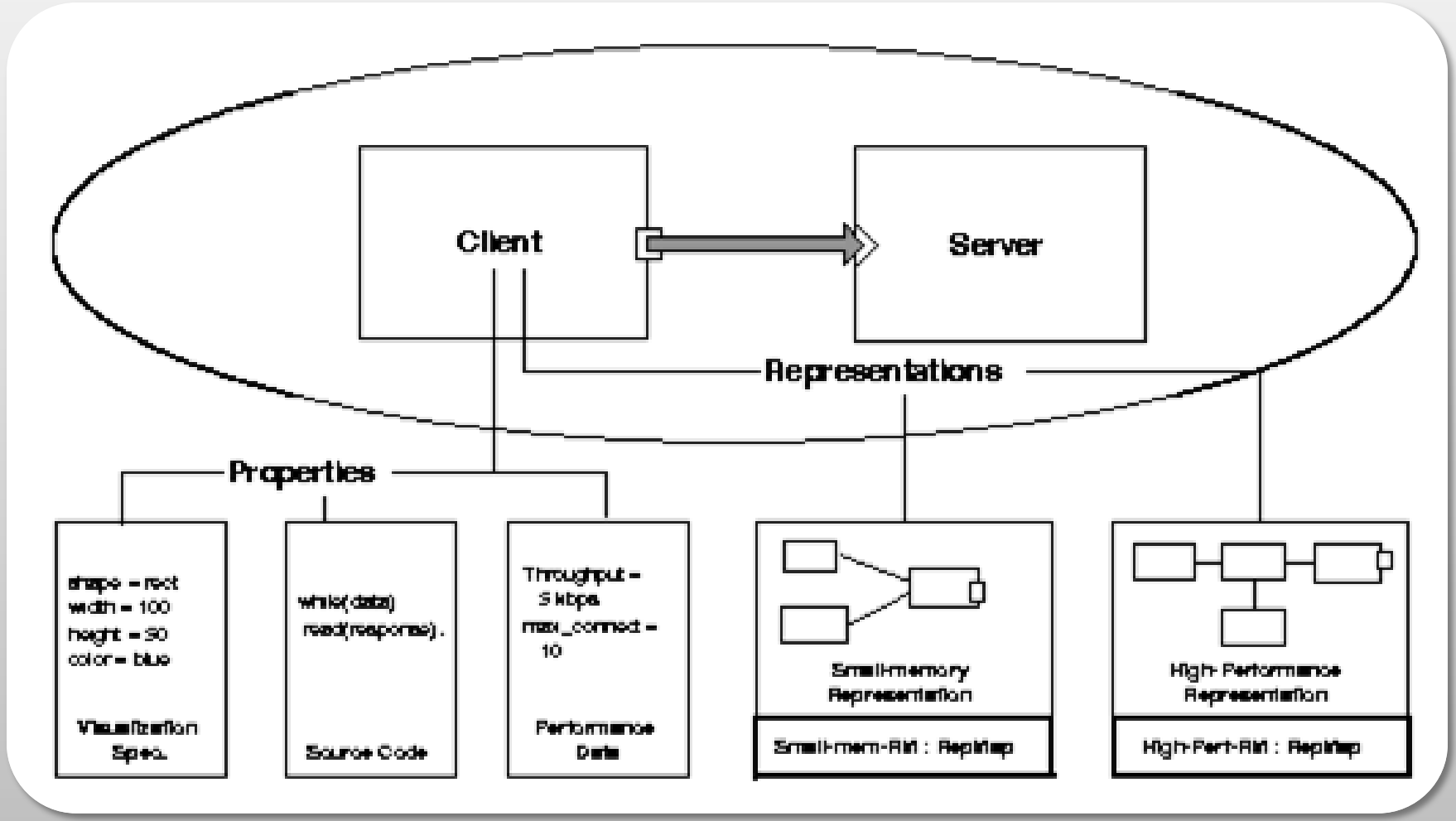


4. ACME PARTICULARITIES (CONT.)

- ACME method allows a translation mechanism…
  - … using properties
    - Runtime semantics
    - Data type for communication between components
    - Protocols of interaction
  - … allowing other tools to interpret the architecture
    - ADL's (Wright, Unicon, etc.)
    - Development environments
    - Analysis or checker tools
- Type structuring allows the architect to create templates to be
  - Used within a project
    - from one client/one server to multiple clients/multiple servers architecture
  - Reused in other projects involving the same kind of structure

4. ACME PARTICULARITIES (CONT.)

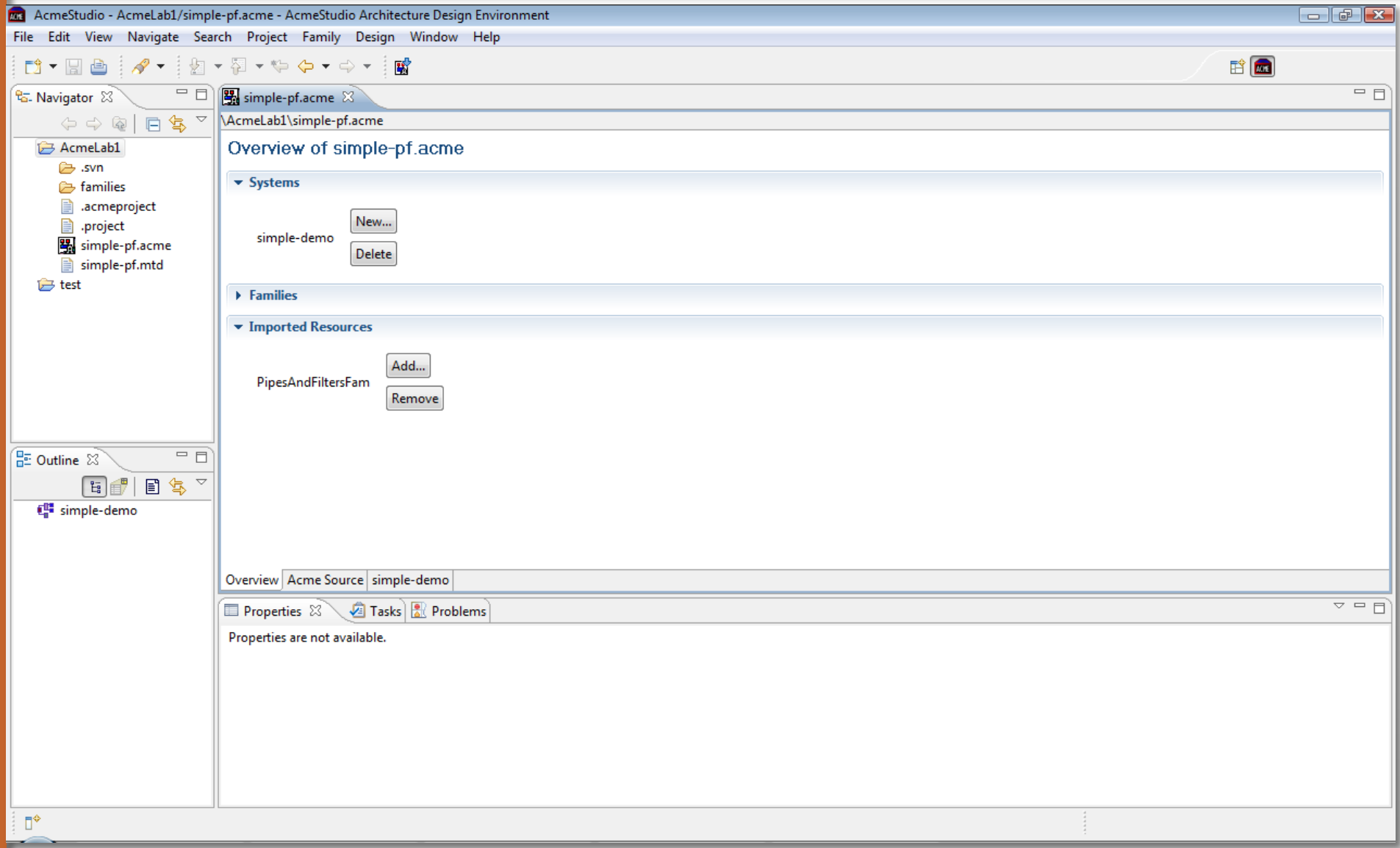
- Illustration of **representation** and **properties** of a component



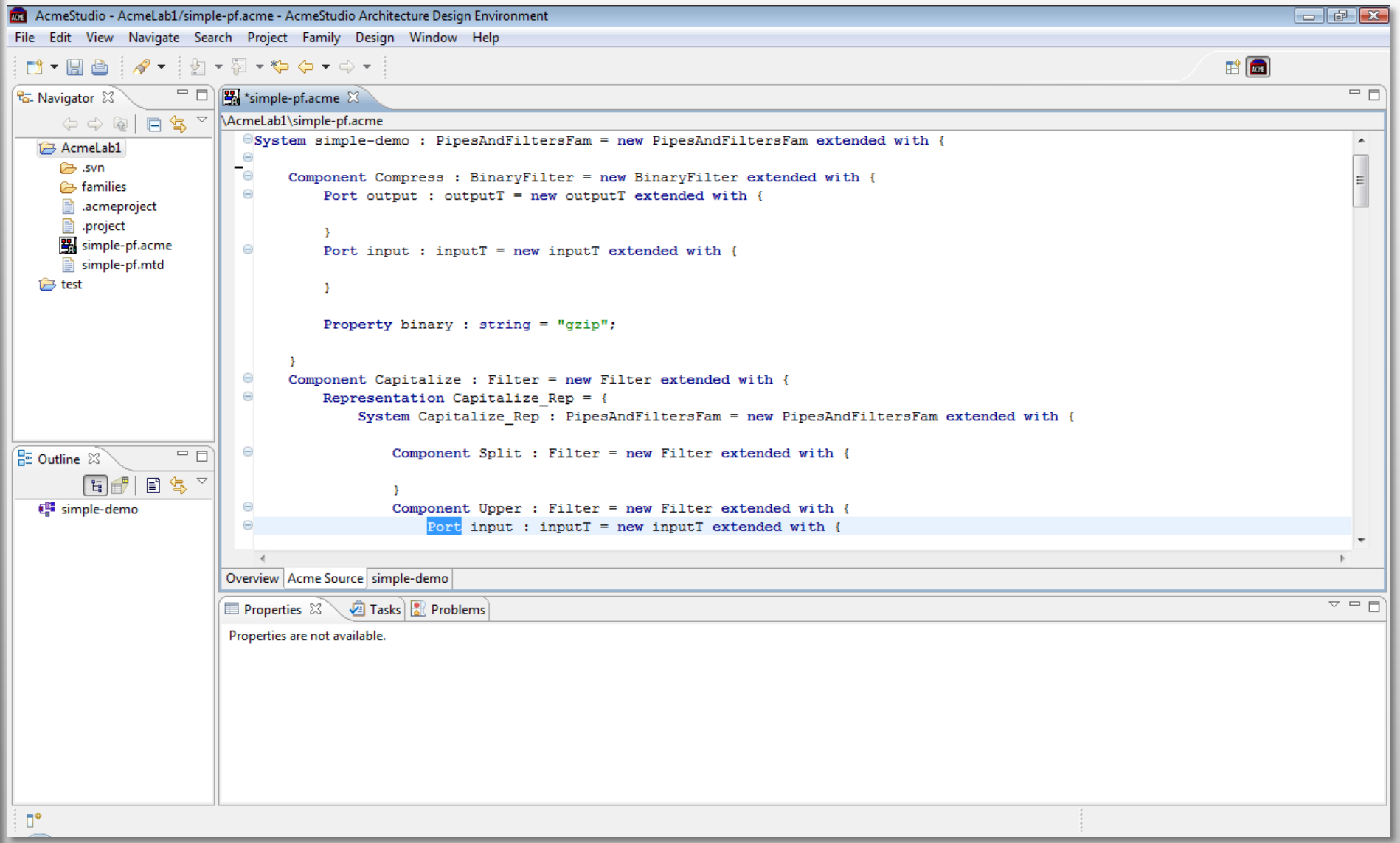
5. ACME TOOL

- ***Eclipse*** plug-in software : ***AcmeStudio***
- ***AcmeStudio***
  - Graphical interface
  - Architecture drawing
  - Design analyze
  - Language description (development)
- Features
  - Create or edit families
  - Edit visualization
  - Edit and check rules
  - Edit properties
  - Etc…

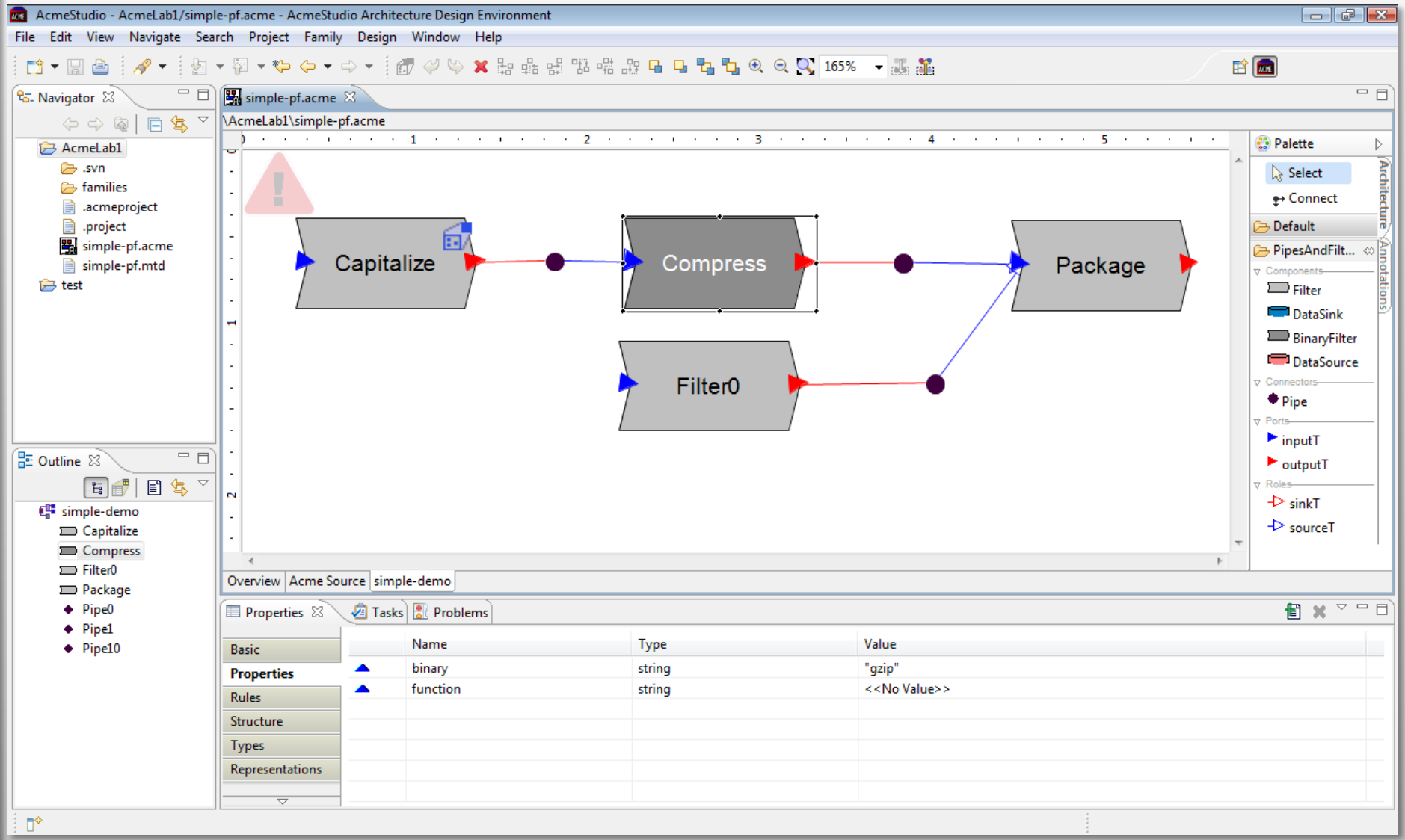
5. ACMESTUDIO



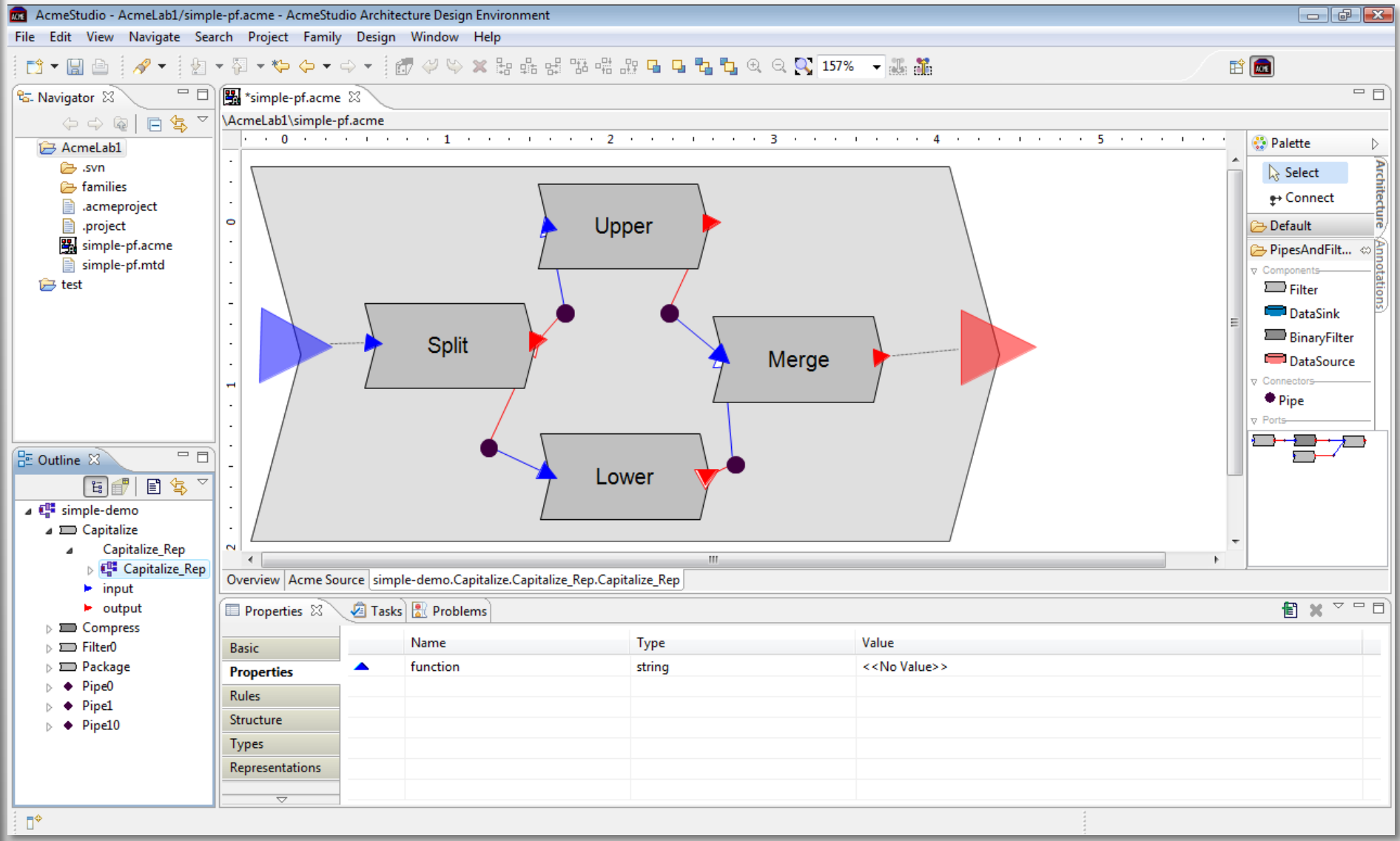
5. ACMESTUDIO (CONT.)



5. ACMESTUDIO (CONT.)



5. ACMESTUDIO (CONT.)



6. ADLS COMPARISON : THE SCOPE

- Comparison of scope

Scope				
ACME	Darwin	Rapide	Unicon	Wright
Architectural interchange, predominantly at the structural level	Architectures of highly-distributed systems whose dynamism is guided by strict formal underpinnings	Modeling and simulation of the dynamic behavior described by an architecture	Data-flows architectures with high volume of data and real-time requirements.	Modeling and analysis of the dynamic behavior of concurrent systems

6. ACME : QUALITIES AND LACKS

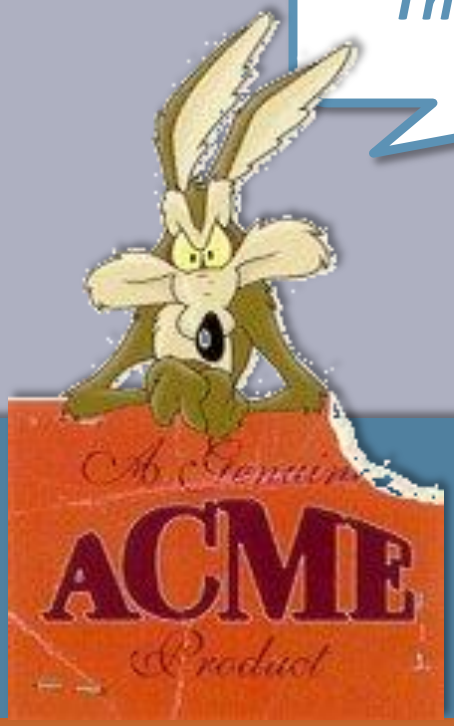
- A common interchange language:
  - Provide variety of tools belonging to several ADLs
  - Easy to use : one format for all ADLs, programmers don't have to master all ADL's languages
- User-friendly interface
  - Very complete, lot of functions
  - Seven basics entities
  - Easy for the user, don't need to learn ACME language
- Group each auxiliary information from ADLs, by using properties
  - Ex : Property Aesop-style : style-id = clientserver;

6. ACME : QUALITIES AND LACKS (CONT.)

- Acme provides translation between 2 ADLS
  - → don't provide advanced tools for each ADL
  - → you may turn to another tool if you want describe detailed architecture
- Should limit the class of systems of ADLs for translations
  - often important and painful trade-offs have to be made to permit the success of translation
- Try to develop a translator bi-directionality will complicate the program a lot
  - → loss of a main functionality

ACME

A powerful ADL



Thankyou for attention !

Project 2
OP4

ADL presentation

Euro Team

Alauzet Pierre, Ahvenniemi Mikko,

Colin Julien, Starck Benoit

KAIST

REFERENCES

1. **David S. Wile,** *ACME: An Interchange Language for Architecture Representation*

2. http://www.cs.cmu.edu/~acme/, *ACME websites*

3. http://en.wikipedia.org/wiki/Architecture_description_language, *ADL wikipedia page*

4. **[KKC00] R. Kazman, M. Klein, P. Clements**, *ATAM: A Method for Architecture Evaluation, CMU/SEI-2000-TR-004, Software Engineering Institute, Carnegie Mellon University, 2000.*