TDDE41 Software Architectures Design and Visualisation

Lena Buffoni
lena.buffoni@liu.se

LINKÖPING UNIVERSITY

Lecture plan

- Architectural views and modeling
- Visualization formalisms and tools
- Designing in an agile context

Views: A building model

Tax collectors view

Renters view

Electricians view

Builders view

Realtors view

Designers view

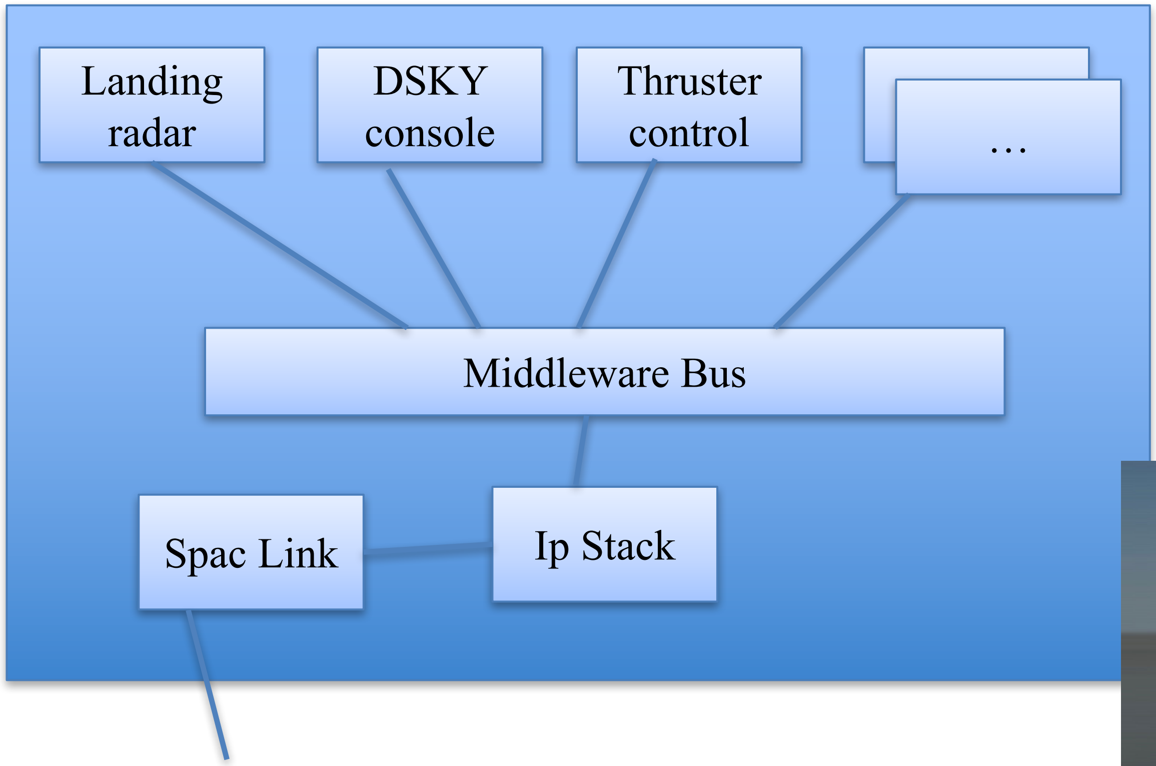
Each view concentrates on modeling specific aspects

Views: a software model

- Logical
- Physical
- Deployment
- Concurrency
- Behavioral

Possible inconsistencies:
behavioral: the system should be robust
physical: the system is implemented with a single server

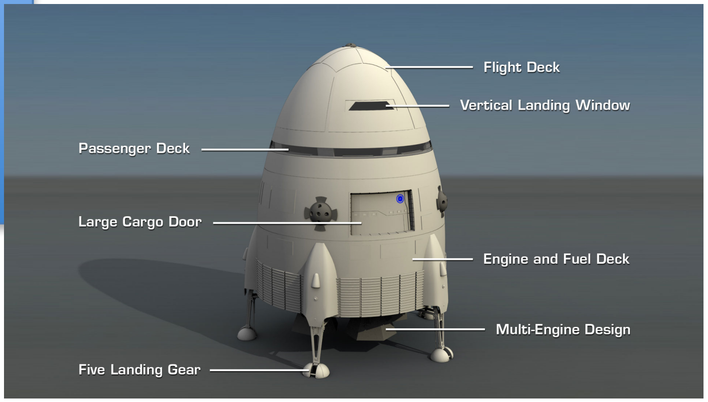
Example: Lunar Lander



Ground System

Logical view

Physical



LINKÖPING UNIVERSITY

*image from https://gatewayspaceport.com

Natural language & informal notations

- The lunar lander has three components: a data storage, a calculation unit and a UI
- The data storage contains height, velocity and fuel data and current simulator time
- The calculation component gets height, velocity and fuel from data storage, updates them with respect to burn rate and returns them
- The UI displays the current status and information

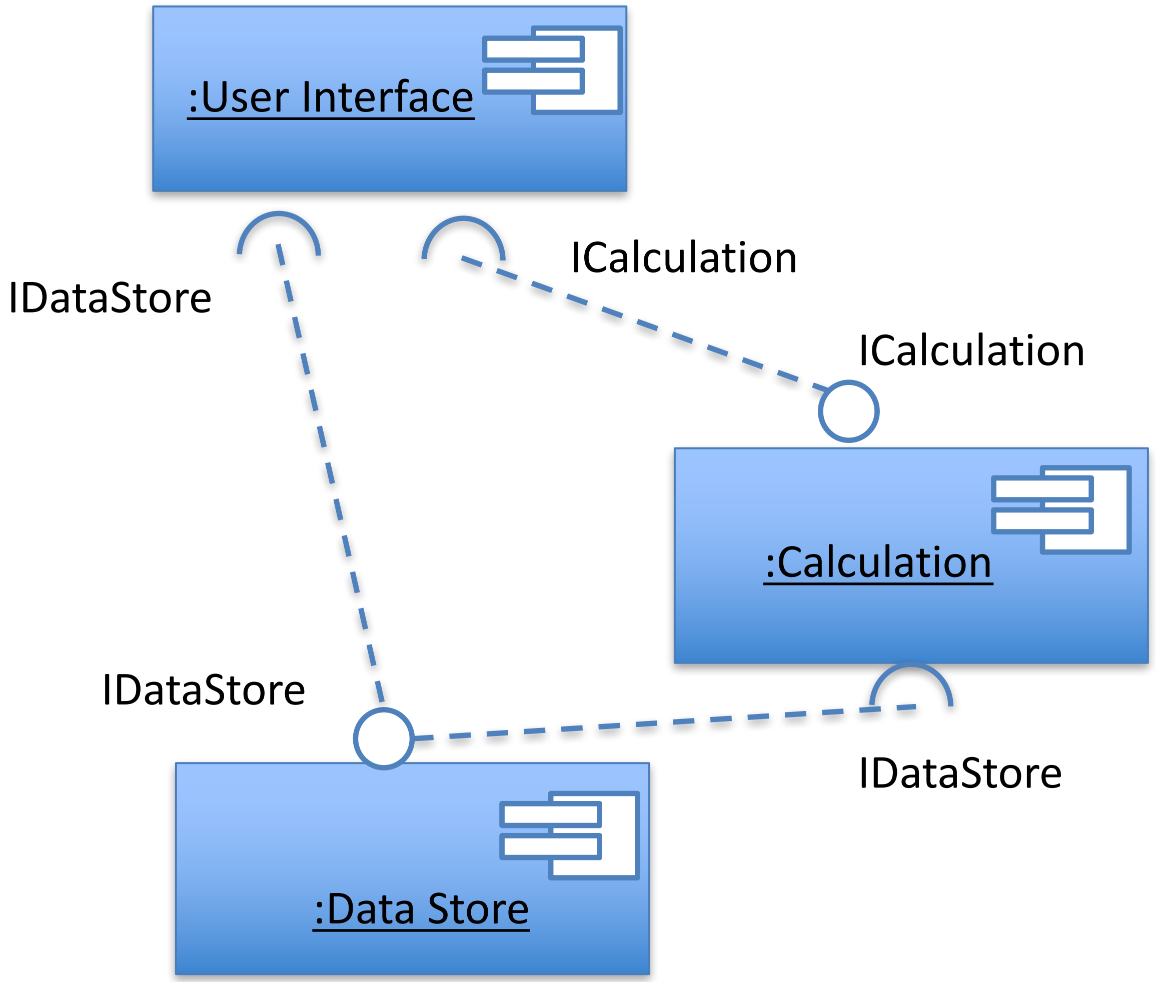
Pros and Cons

+ good for non functional properties
+ a good complement to more formal specifications

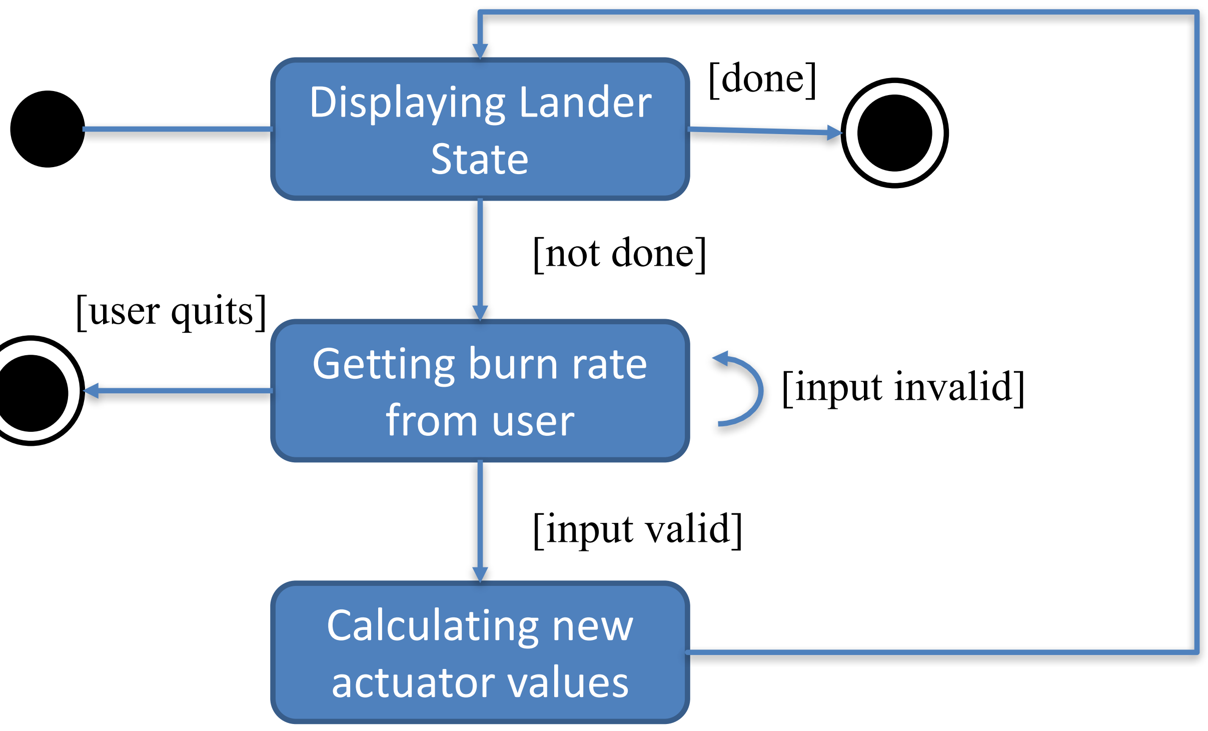
But

- Ambiguous
- Difficult to get an overview of the problem
- Often incomplete

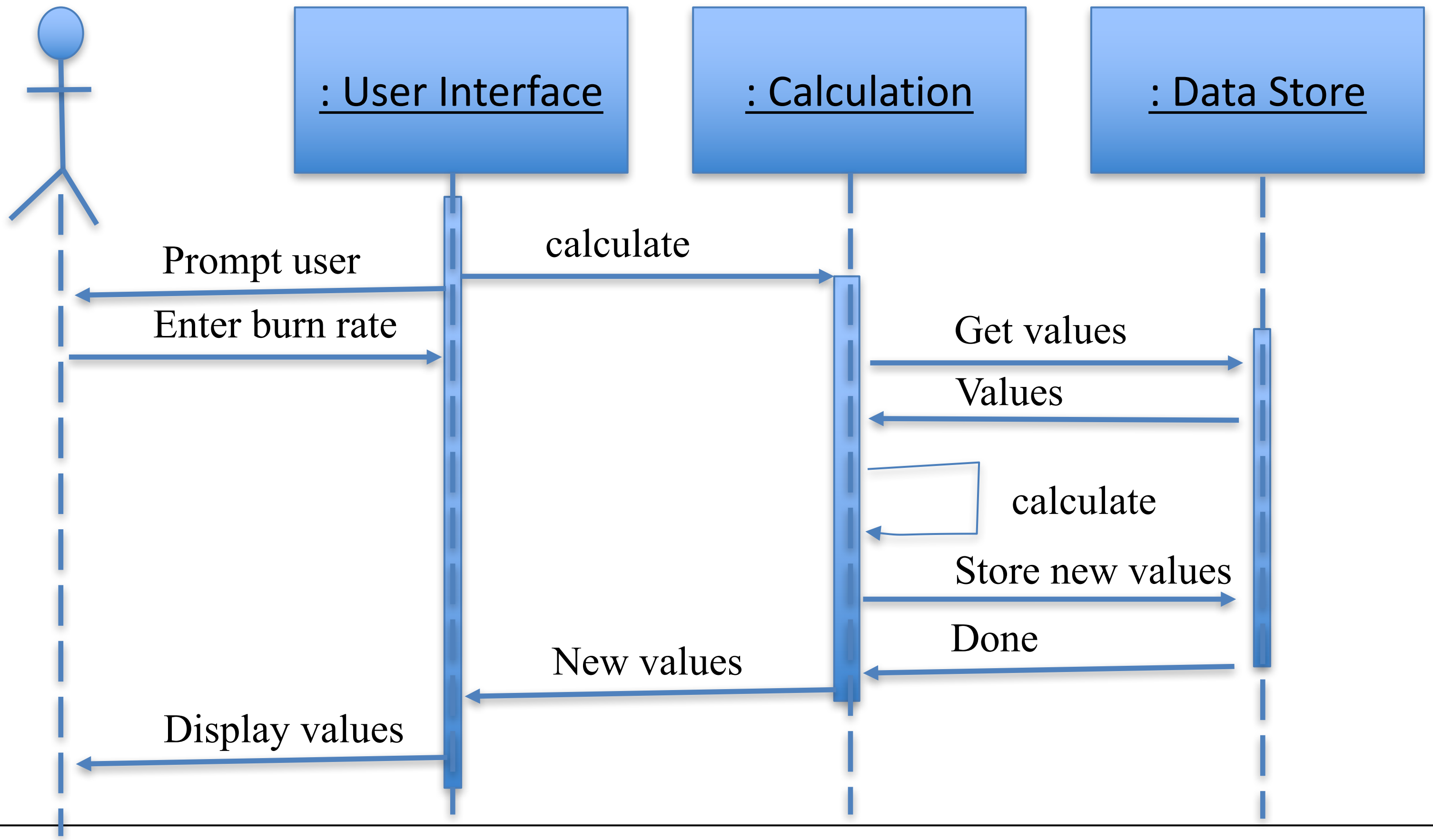
UML – component diagram



UML - statecharts



UML – sequence diagram



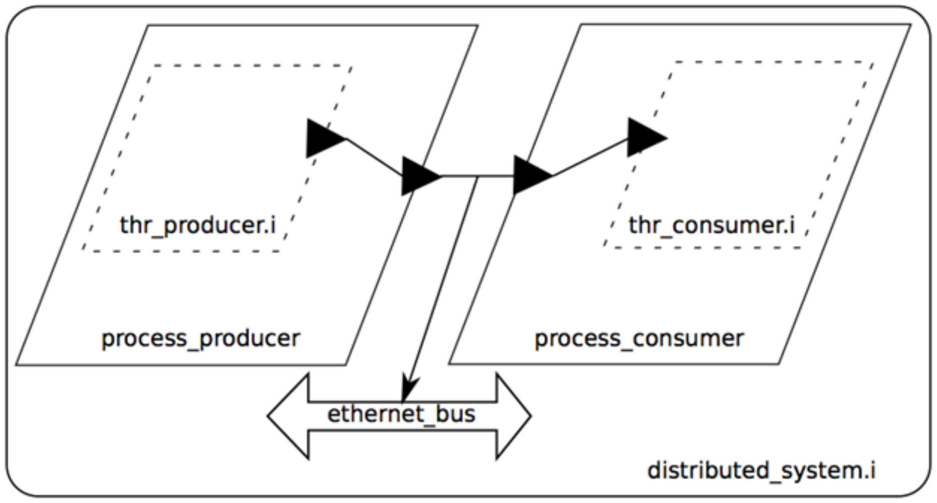
Pros and Cons

+ Can express a lot of concepts
+ Extensive tool support
- Open to ambiguities

-> Good practice is to develop specialized profiles

Architecture Analysis and Design Language (AADL)

- Initially for modelling avionic systems
- Language for system architectures
- Component based: type and implementation



AADL: lunar lander calculation system

```
system calculation_type
     features
     network : requires bus access lan_bus.calculation_to_datastore;
     request_get : out event port;
     response_get : out event port;
     request_store : out event port lander_state_data;
     response_store : in event port;
end calculation type;

system implementation calculation_type.calculation
subcomponents
     the_calculation_processor : processor calculation_processor_type;
     the_calculation_process: process calculation_process_type.one_thread;
connections
     bus acces network -> the_calculation_processor.network;
     event data port response_get -> the_calculation_process.response_get;
     …
properties
     Actual_Processor_Binding => reference the_calculation_processor
     applies to the_calculation_process;

end calculation_type.calculation;
```

LINKÖPING
UNIVERSITY

Pros and cons

+ Supports different types of analysis
+ Good for critical systems

- Complex

xADL: Extensible XML-based ADL

- Promote feature reuse
- Facilitate addition of new features
- Relies on XML for extensibility
- A composition of different schemas covering different aspects
- Supported by a variety of tools for visualization and consistency verification
- Provides a xADL data binding library in Java

Lunar Lander in xADL

```
xArch{

     archStructure{
     id = "lunarlander";
     description = "Lunar Lander";

     component{
     id = "calculation";
     description = "Calculation";
     interface{
     id = "calculation.getValues";
     description = "Calculation Get Values interface";
     direction = "out";}
     …}
     link{
     id = "calculation-to-datastore-getvalues";
     decription = "calculation to data store get values";
     point{

     anchorOnInterface{
     type = "simple";
     href = "#datastore.getValue"
     }}
…

}}
```

Static and dynamic aspects

- Static: do not involve behaviors during runtime

- Dynamic: changes to the structure during runtime – eg: component failures, dynamic connections

Visualization types

graphical

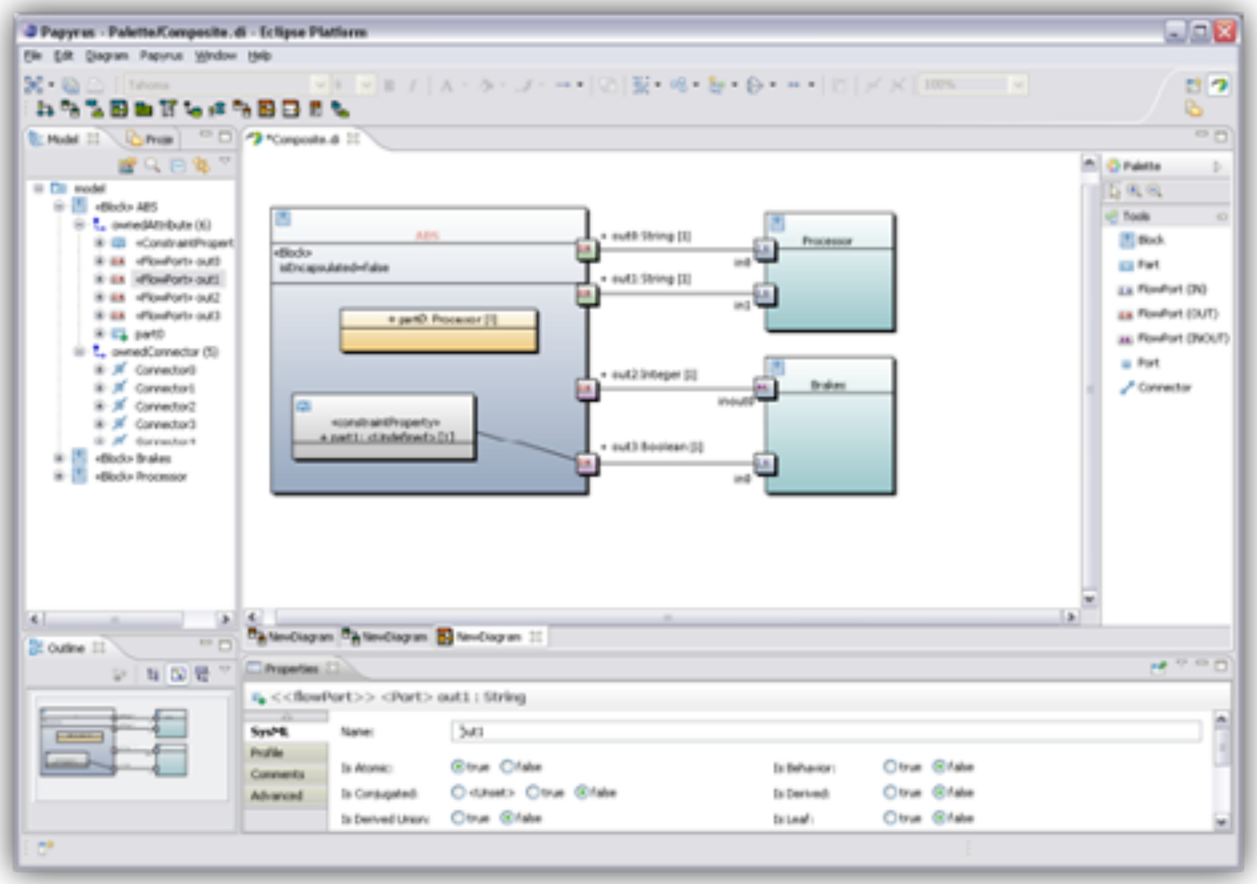
UML diagrams

text

hybrid

xADL description

UML: Papyrus



<<flowPort>> <Port> out1 : String

Name: out1

Is Atomic: [✓ true] false

Is Conjugated: <Unset> true [✓ false]

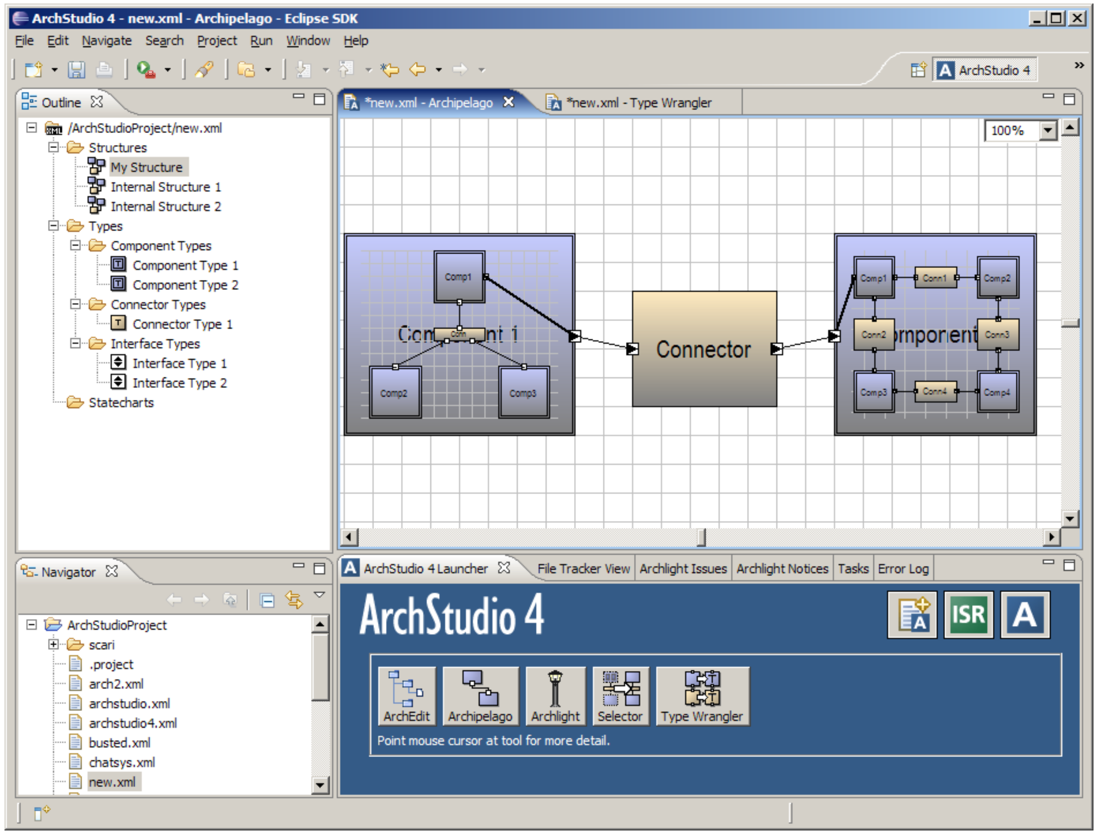
Is Derived Union: true [✓ false]

Is Behavior: true [✓ false]

Is Derived: true [✓ false]

Is Leaf: true [✓ false]

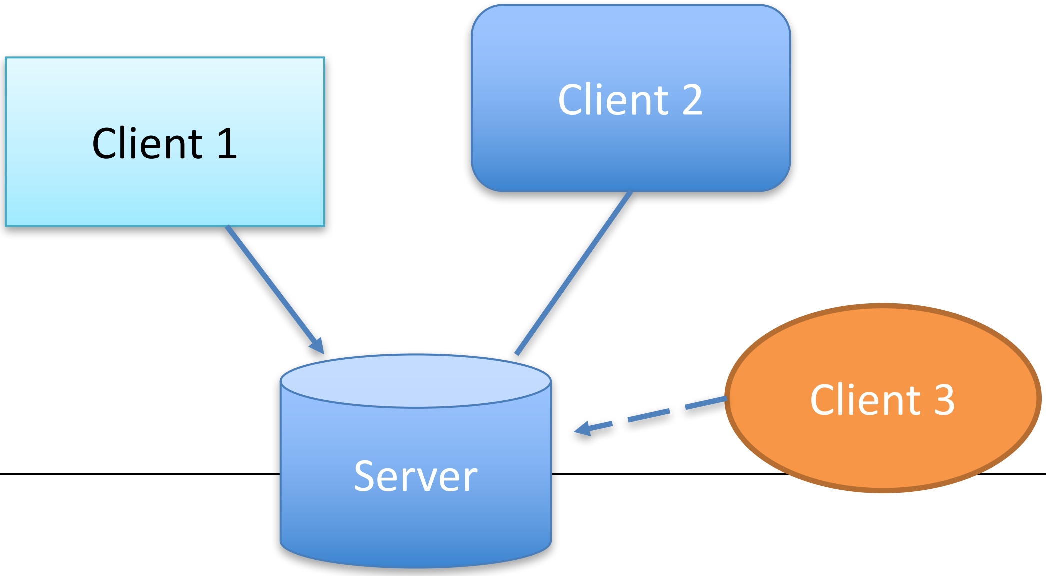
x:ADL - ArchStudio



Common issues

- Same symbol – different meaning
- Differences without meaning
- Decorations without meaning
- Borrowed symbol – different meaning

Coordinating visualizations!



Choosing a visualisation

- Fidelity
- Consistency
- Comprehensibility
- Dynamism
- View coordination
- Aesthetics
- Extensibility

Attention: Distinction between language features and editor features

Discussion

How do agile methods impact the architecture development process?

Agile methods and architecture

How to document something that is constantly changing?

eg: a browser automatically downloading plugins

- Document what is true about all versions of the system
- Document how the system is allowed to change

Agile working methods

- Agile =/= no templates
- Add information on a “as needed” basis
- Do not spend time filling in information not needed now

Discussion

Architectural design in uncharted territory?

Summary

Documentation is needed to:

- Communicate with stakeholders
- Analyze the architecture
- Learn from the architecture

The End.

Questions?

www.liu.se

LINKÖPING
UNIVERSITY