Architecture Description Languages & Tools

Carlos Diego Damasceno

Milena Guessi

SCC-5944 Software Architecture – 2016

Profa. Dra. Elisa Yumi Nakagawa

Program

1. History
2. Architecture modeling elements
3. ISO/IEC/IEEE 42010
4. Formalism levels
5. Examples
6. Tools
7. State-of-the-practice

History

Traditional Definitions

- Provide mechanisms for expressing composition, abstraction, reusability, configuration, and analysis of software architectures (Shaw and Garlan, 1994)
- An ADL must explicitly model components, connectors, and their configurations; furthermore, to be truly usable and useful, it must provide tool support for architecture-based development and evolution (Medvidovic and Taylor, 2001)

Architecture modeling elements

Characteristics

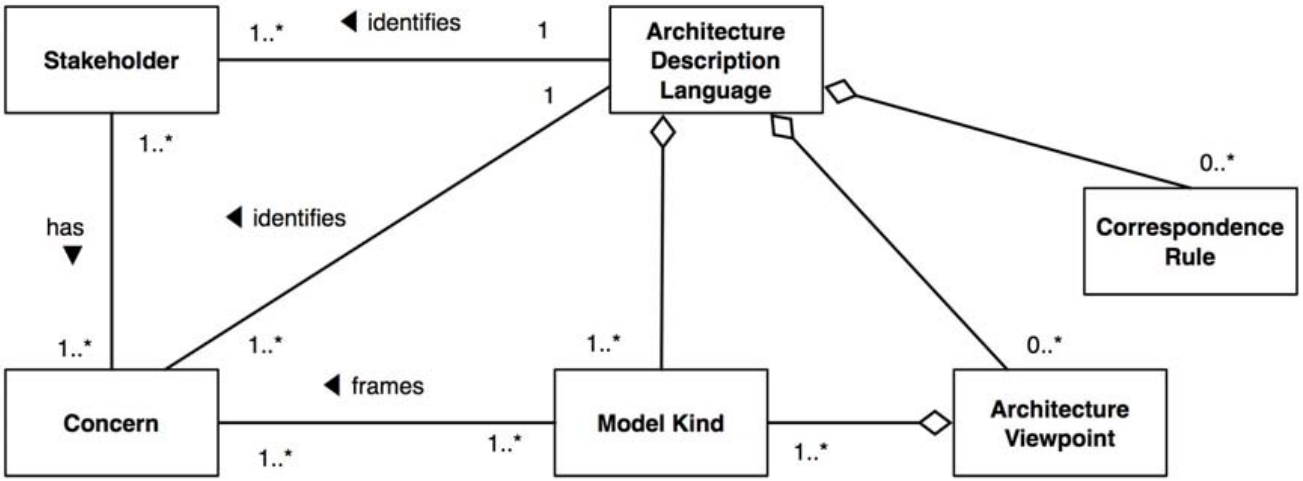
- Architecture building blocks
  - Components
  - Connectors
  - Configurations
- Tool Support
  - Automated analyses on the architecture description

Architecture modeling elements

- Components and Connectors
  - Interface
  - Type
  - Semantics
  - Constraints
  - Evolution
  - Non-functional properties
- Tool Support
  - Active specification
  - Multiple views
  - Analysis
  - Refinement
  - Implementation generation
  - Dynamism
- (Architectural) Configuration
  - Understandability
  - Compositionality
  - Refinement and traceability
  - Heterogeneity
  - Scalability
  - Evolution
  - Dynamism
  - Constraints
  - Non-functional properties

Source: Medvidovic, N. and Taylor, R. N., A Classification and Comparison Framework for Software Architecture Description Languages, IEEE Trans. Softw. Eng. 26, 1, 70-93, 2000.

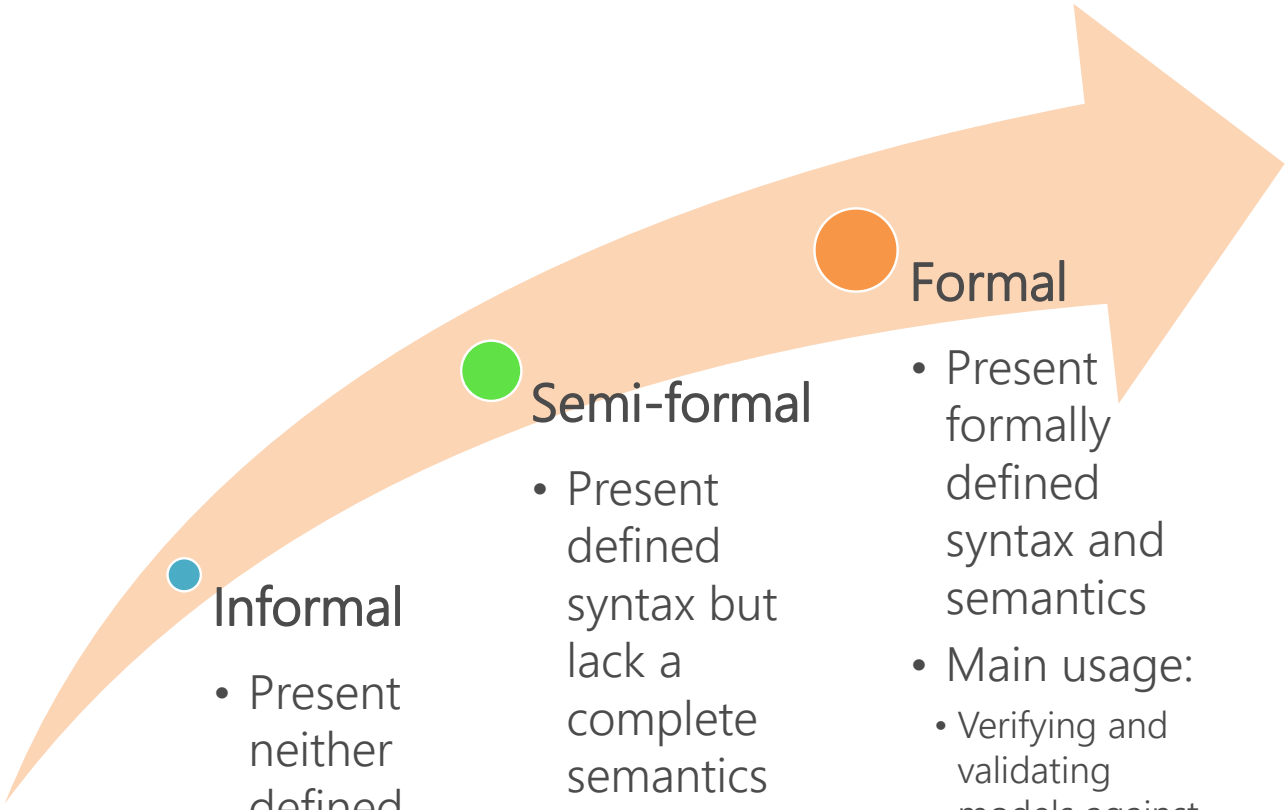
ISO/IEC/ IEEE 42010



Conceptual model of an architecture description language

Source: ISO/IEC/IEEE 42010:2011 Systems and software engineering - Architecture description, p.1-46, Dec. 1 2011.

Formalism levels



Informal

- Present neither defined syntax or semantics
- Main usage:
  - Illustrating or exemplifying concepts

Semi-formal

- Present defined syntax but lack a complete semantics
- Main usage:
  - Supporting communication among stakeholders

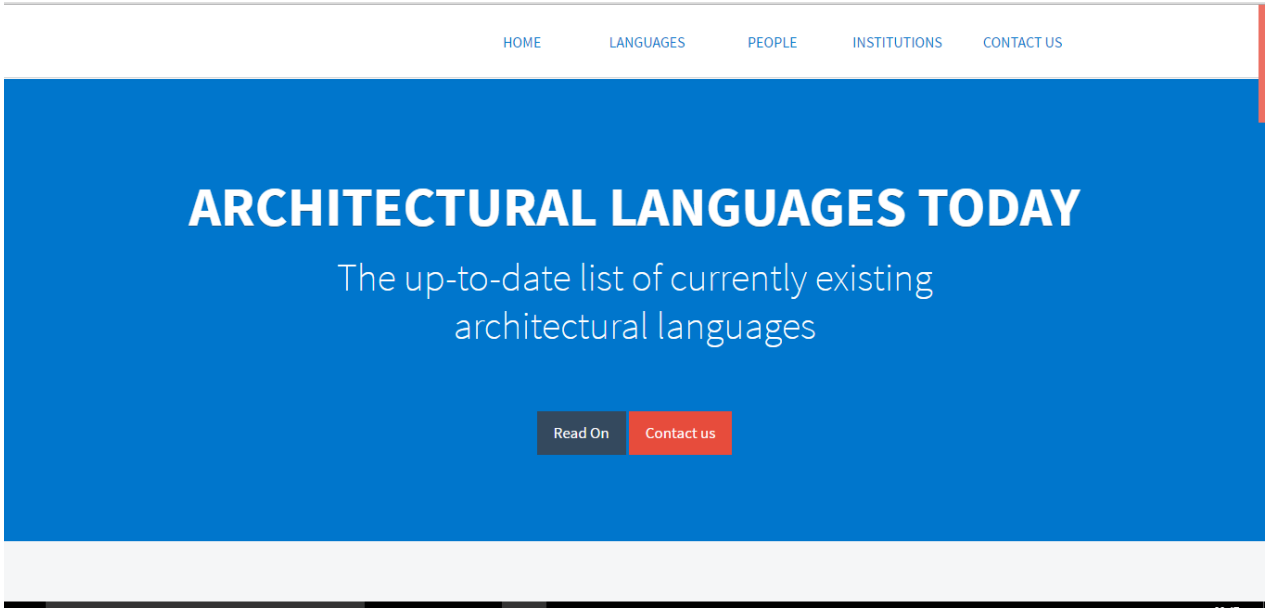
Formal

- Present formally defined syntax and semantics
- Main usage:
  - Verifying and validating models against properties and quality attributes

Examples

Examples

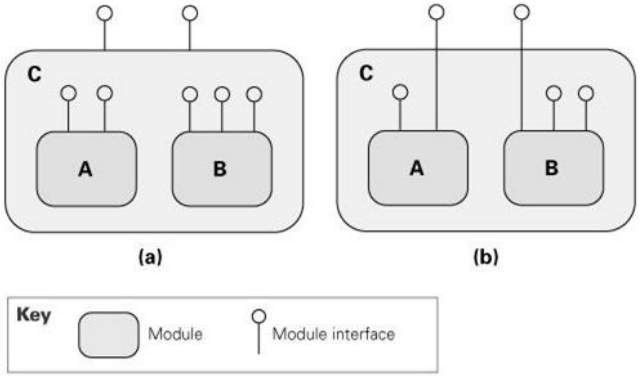
- Many, many, many ADLs…
  - 123!!



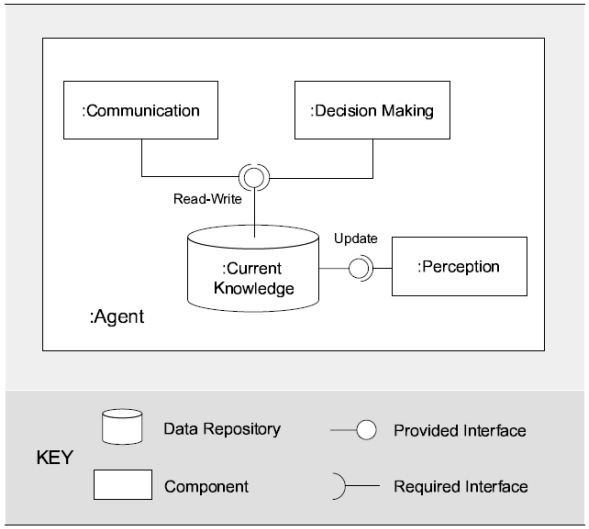
Source: http://www.di.univaq.it/malavolta/al/

Examples

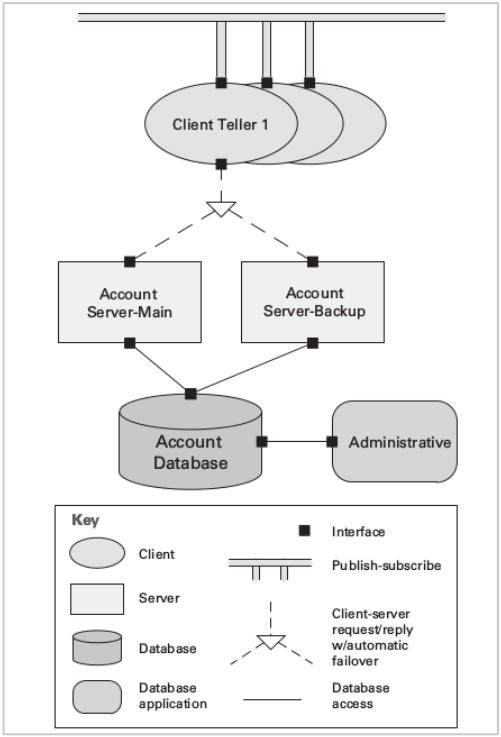
Informal



*1. Modules can **(a)** provide interfaces, hiding other modules, or **(b)** exposing some interfaces of internal modules*



3. Shared data view of an agent

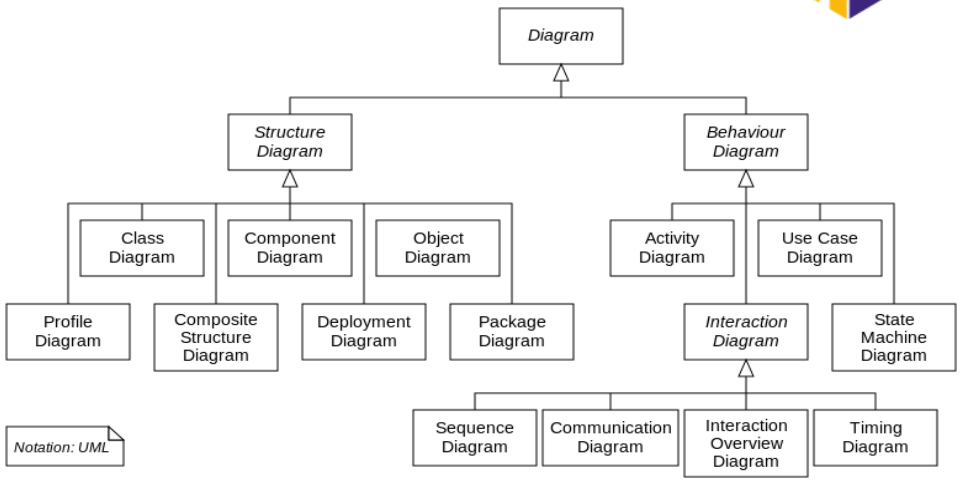


2. A bird's-eye-view of a system as it appears at run-time.

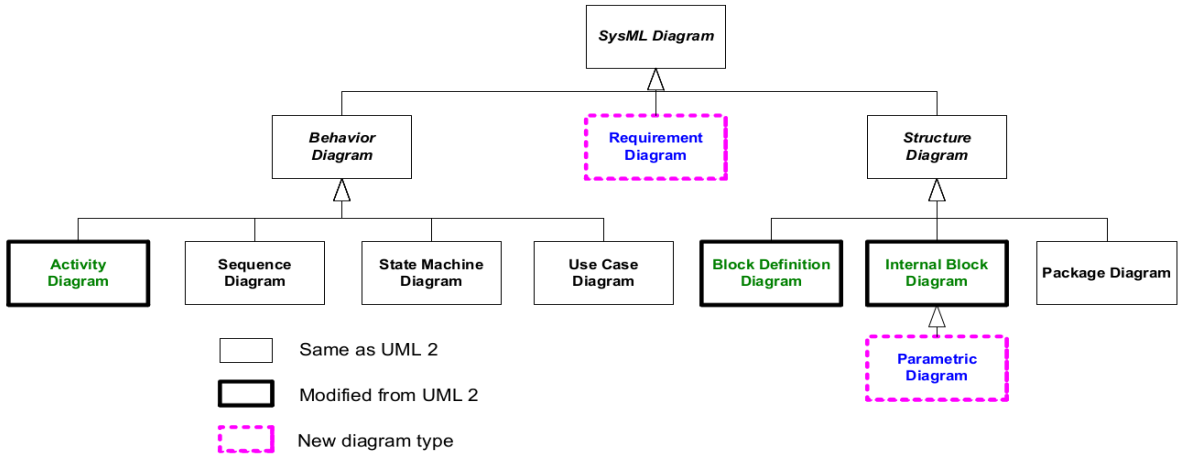
Source:
1,2 Clements, P. et al. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2011
3 Weyns, D. An Architecture-Centric Approach for Software Engineering with Situated Multiagent Systems. PhD Thesis. 2006. Available at: http://www.cs.kuleuven.be/publicaties/doctoraten/cw/CW2006_09.abs.html

Examples

Semi-formal



1. UML 2.x diagram types

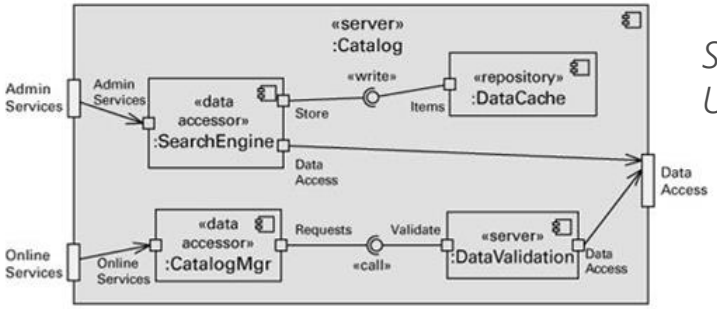


2. SysML 1.x diagram types

Source:
1 http://www.omg.org/spec/UML/2.5/
2 http://www.omg.org/spec/SysML/1.4/

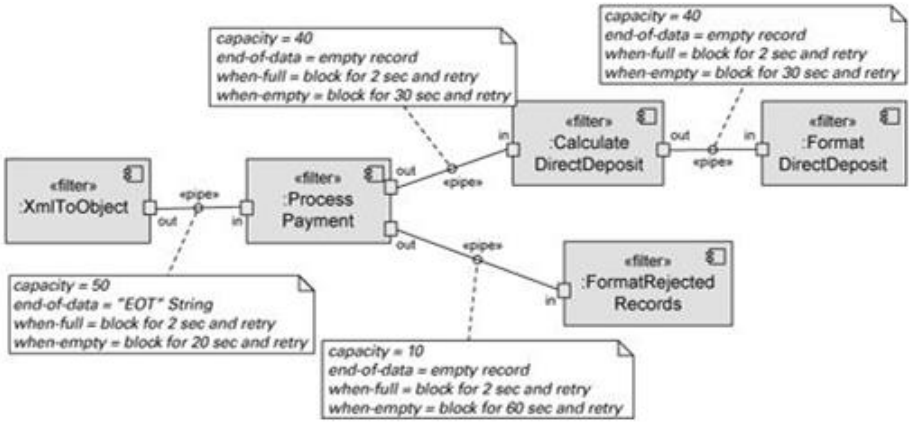
Examples

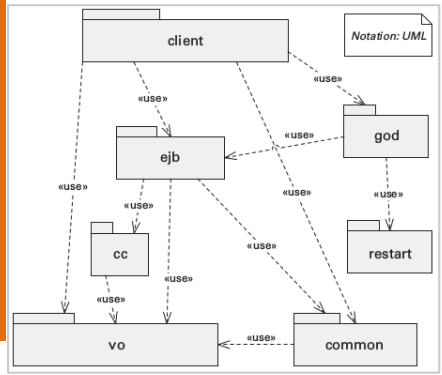
Semi-formal UML



Substructure of a UML component

UML diagram of a pipe-and-filter view





using module / used module	client	ejb	cc	god	restart	common	vo
client	0	0	0	0	0	0	0
ejb	1	0	0	1	0	0	0
cc	0	1	0	0	0	0	0
god	1	0	0	0	0	0	0
restart	0	0	0	1	0	0	0
common	1	1	0	0	0	0	0
vo	1	1	1	0	0	1	0

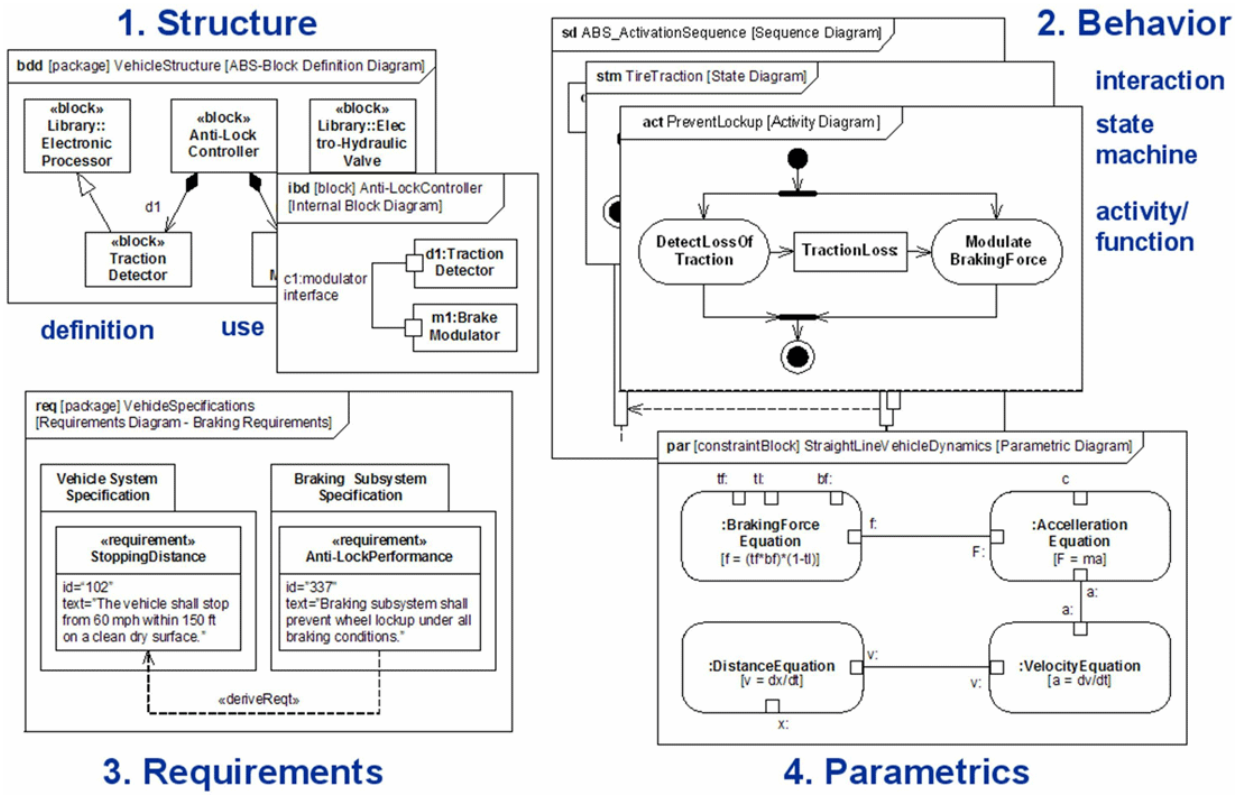
Key: "1" means module in column uses module in row

UML package diagram (left) and Dependency Structure Matrix (DSM) (right)

Source: Clements, P. et al. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley, 2011

Examples

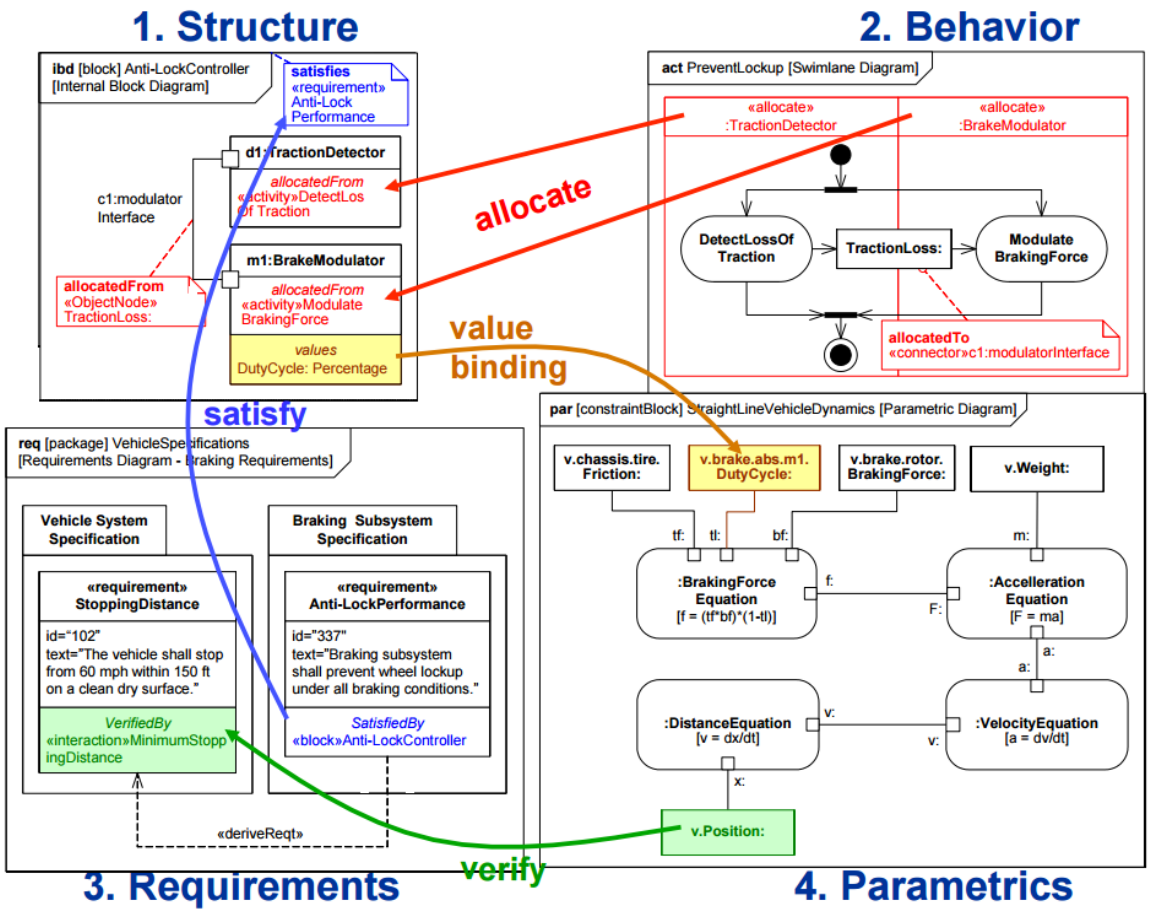
Semi-formal SysML



Source: http://www.omgsysml.org/

Examples

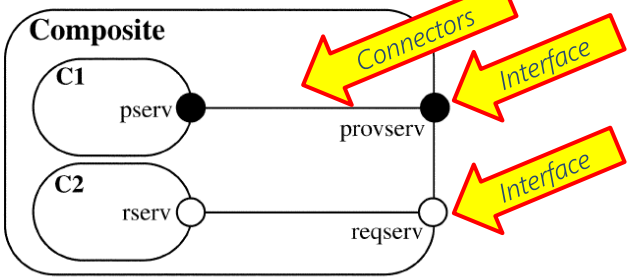
Semi-formal SysML



Source: http://www.omgsysml.org/

Examples
Formal

```
component Composite
  provide provserv;
  require reqserv;
  inst
    C1 : CompType1;
    C2 : CompType2;
  bind
    provserv -- C1.pserv;
    C2.rserv -- reqserv;
}
```



A composite component specified in *Darwin* (top) and (bottom) the graphical view of the component

```
Sample_Arch.addComponent(Comp5);
Sample_Arch.weld(Conn1, Comp5);
Sample_Arch.weld(Comp5, Conn2);
Comp5.start();
```

Dynamic insertion of a component into a *C2SADEL* architecture.

```
Style Pipe-Filter
  . . .
  Constraints
      ∀ c : Connectors • Type(c) = Pipe
  ∧   ∀ c : Components; p : Port | p ∈ Ports(c) •
        Type(p) = DataInput ∨ Type(p) = DataOutput
```

The pipes-and-filters style declared in *Wright*.

```
Family fam = {
  Component Type comp1 = { Port p1; }
  Component Type comp2 = { Port p2; }
  Connector Type conn1 = { Roles r1, r2; }
}

Family sub_fam extends fam with {
  Component Type sub_comp1 extends comp1 with {
    Port p1 = { Property attach : int <<default = 1>>; }
  }
  Component Type comp3 = { ... }
}
```

Declaration in *ACME* of a family of architectures, ***fam***, and its subfamily, ***sub_fam***, which has new components and properties

Source: Medvidovic, N. and Taylor, R. N., A Classification and Comparison Framework for Software Architecture Description Languages, IEEE Trans. Softw. Eng. 26, 1, 70-93, 2000.

Examples

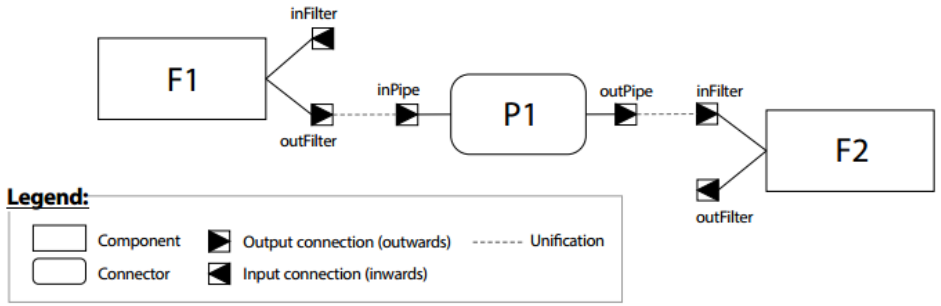
Formal

π-ADL

```
component Filter is abstraction() {
   connection inFilter is in(String)
   connection outFilter is out(String)
   protocol is {
      (via inFilter  receive String
       via outFilter send String)*
   }
   behaviour is {
      transform is function(d : String) : String {
         unobservable
      }
      via inFilter receive d : String
      via outFilter send transform(d)
      behavior()
   }
}
```

```
connector Pipe is abstraction() {
   connection inPipe is in(String)
   connection outPipe is out(String)
   protocol is {
      (via inPipe  receive String
       via outPipe send String)*
   }
   behaviour is {
      via inPipe receive d : String
      via outPipe send d
      behavior()
   }
}
```

```
architecture PipeFilter is abstraction() {
   behavior is {
      compose {
            F1 is Filter()
         and P1 is Pipe()
         and F2 is Filter()
      } where {
         F1::outFilter unifies P1::inPipe
         P1::outPipe   unifies F2::inFilter
      }
   }
}
```



Description of a simple pipeline architecture

Source: Cavalcante, E., Oquendo, F., Batista, T. Architecture-Based Code Generation: From π-ADL Architecture Descriptions to Implementations in the Go Language. ECSA 2015.

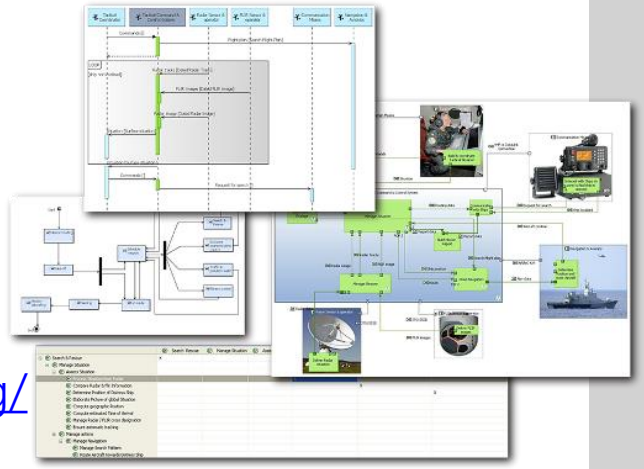
Tools

Tools

PolarSys

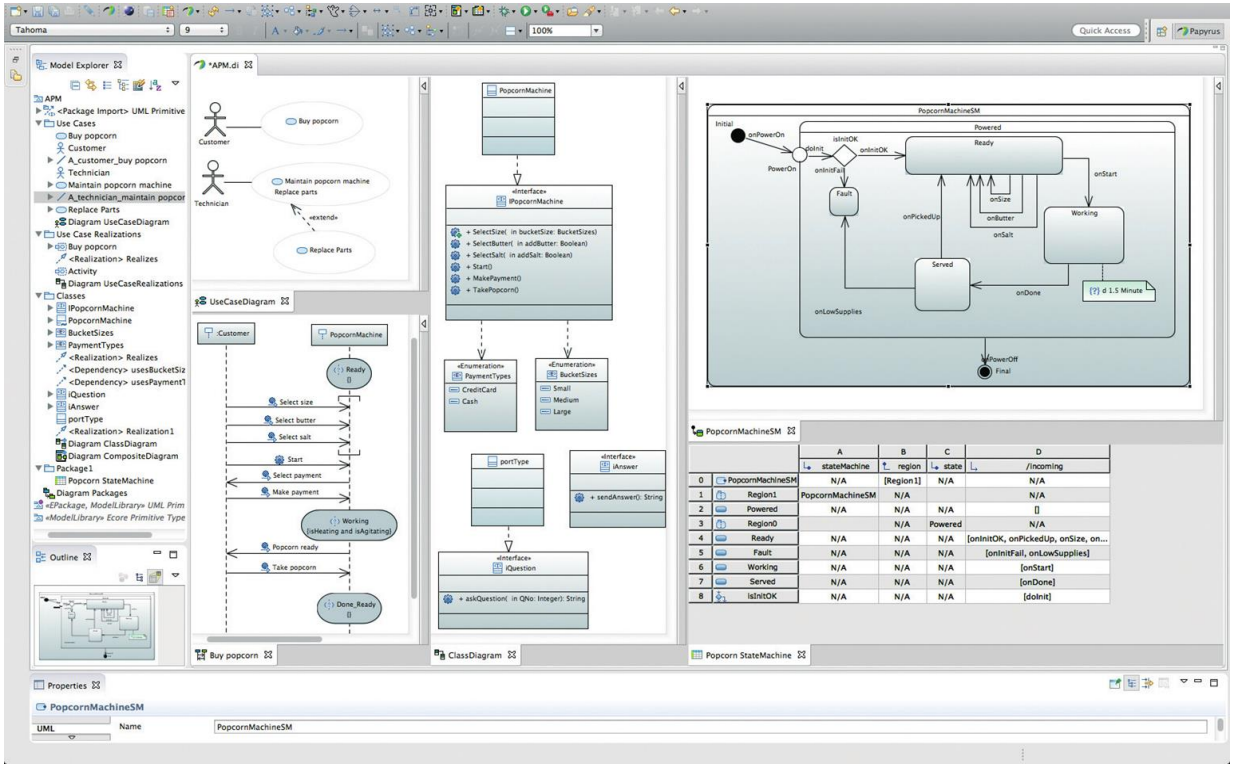
PolarSys
Open Source Tools for Embedded Systems

- Description:
  - Eclipse-based solution for SysML and UML modeling
- Features:
  - Model-based simulation, formal testing, safety analysis, performance/trade-offs analysis, architecture exploration
  - Free and open source
- Support:
  - UML
  - SysML
  - ISO/IEC 42010
- Homepage:
  - https://www.polarsys.org/



PolarSys

Open Source Tools for Embedded Systems



AIRBUS

ERICSSON

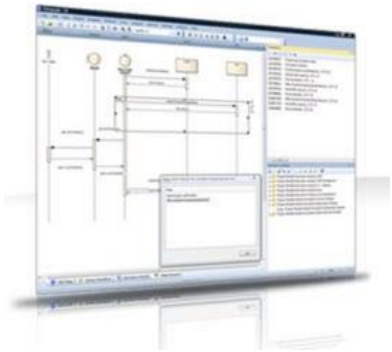
Fraunhofer FOKUS

Atos

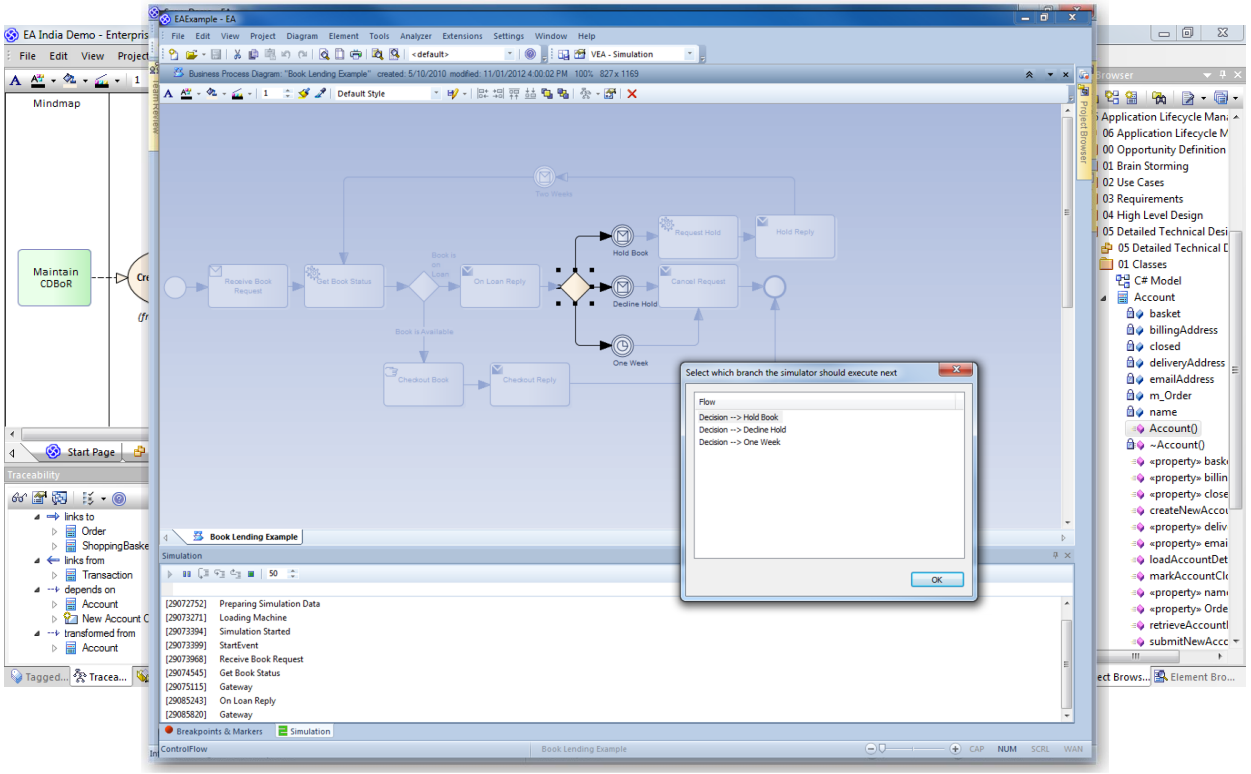
Tools

Enterprise Architect

- Description:
  - High performance modeling, visualization, and design platform based on the UML 2.5
- Features:
  - Business Modeling, Requirements Traceability, Document Generation, Source Code Generation, Reverse Engineering, Systems Engineering and Simulation
  - Trial version (Academic price)
- Support:
  - UML 2.5
  - BPMN
  - SysML
  - MDA
  - C/C++
  - Java



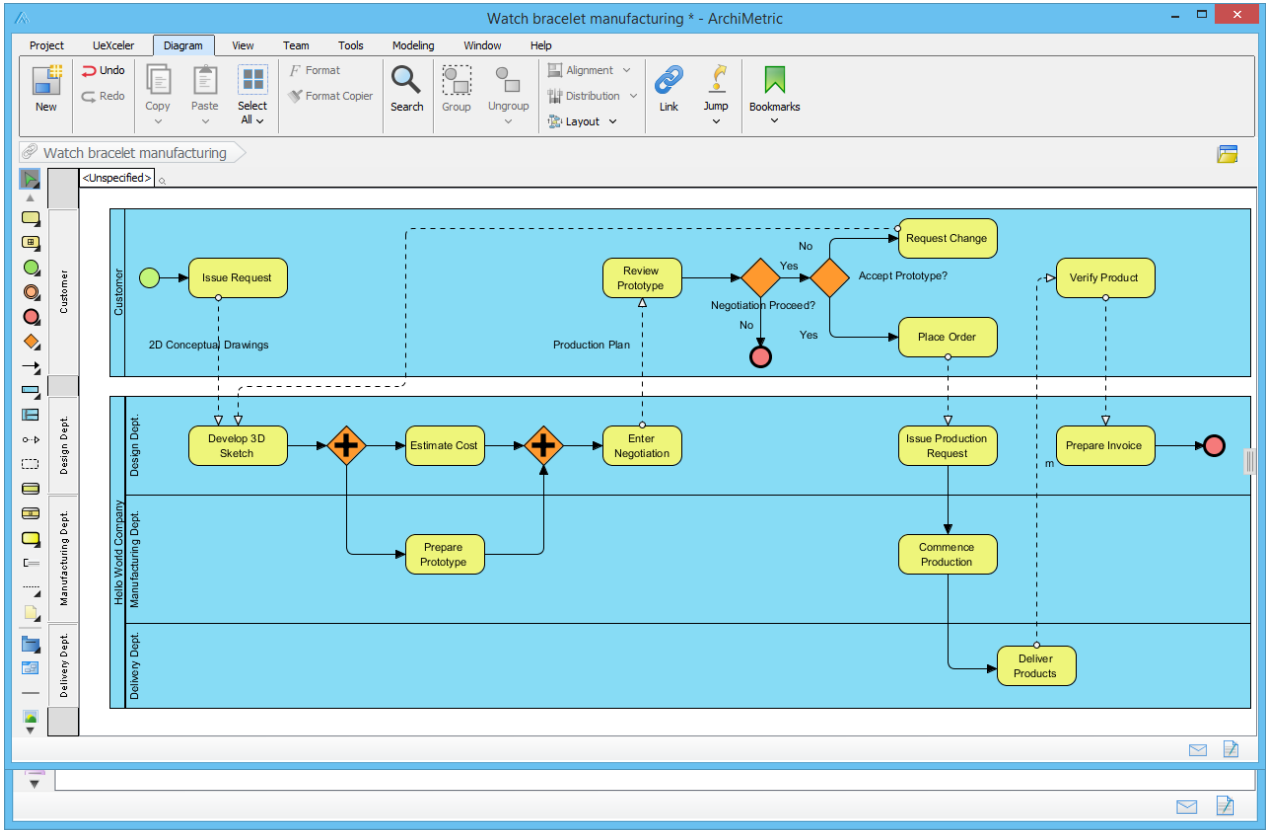
- Homepage:
  - http://www.sparxsystems.com.au/products/ea/index.html


EA India Demo - Enterpris
File Edit View Project
Mindmap
Maintain CDBoR
Start Page
Traceability
links to
Order
ShoppingBaske
links from
Transaction
depends on
Account
New Account C
transformed from
Account
Tagged...
Tracea...
EAExample - EA
File Edit View Project Diagram Element Tools Analyzer Extensions Settings Window Help
VEA - Simulation
Business Process Diagram "Book Lending Example" created: 5/10/2010 modified: 11/01/2012 4:00:02 PM 100% 827x 1169
Default Style
Two Weeks
Receive Book Request
Get Book Status
Book is on Loan
On Loan Reply
Hold Book
Request Hold
Hold Reply
Decline Hold
Cancel Request
One Week
Book is Available
Checkout Book
Checkout Reply
Book Lending Example
Simulation
[29072752] Preparing Simulation Data
[29073271] Loading Machine
[29073394] Simulation Started
[29073399] StartEvent
[29073968] Receive Book Request
[29074545] Get Book Status
[29075115] Gateway
[29085243] On Loan Reply
[29085820] Gateway
Breakpoints & Markers
Simulation
ControlFlow
Book Lending Example
CAP NUM SCRL WAN
Select which branch the simulator should execute next
Flow
Decision --> Hold Book
Decision --> Decline Hold
Decision --> One Week
OK
Browser
Application Lifecycle Mana
06 Application Lifecycle M
00 Opportunity Definition
01 Brain Storming
02 Use Cases
03 Requirements
04 High Level Design
05 Detailed Technical Desi
05 Detailed Technical D
01 Classes
C# Model
Account
basket
billingAddress
closed
deliveryAddress
emailAddress
m_Order
name
Account()
~Account()
«property» baske
«property» billin
«property» close
createNewAccou
«property» deliv
«property» emai
loadAccountDet
markAccountCl
«property» nam
«property» Orde
retrieveAccountI
submitNewAccc
ect Brows...
Element Bro...

Tools

ArchiMetric

- Description:
  - An all-in-one software and system development tool for end-to-end IT system modeling
- Features:
  - Enterprise Modeling, Document Production, Project Management
  - Full-featured trial for 30 days
- Support:
  - UML
  - SysML
  - ArchiMate
  - Entity Relationship Diagram (ERD)
  - Data Flow Diagram (DFD)
- Homepage:
  - http://www.archimetric.com/

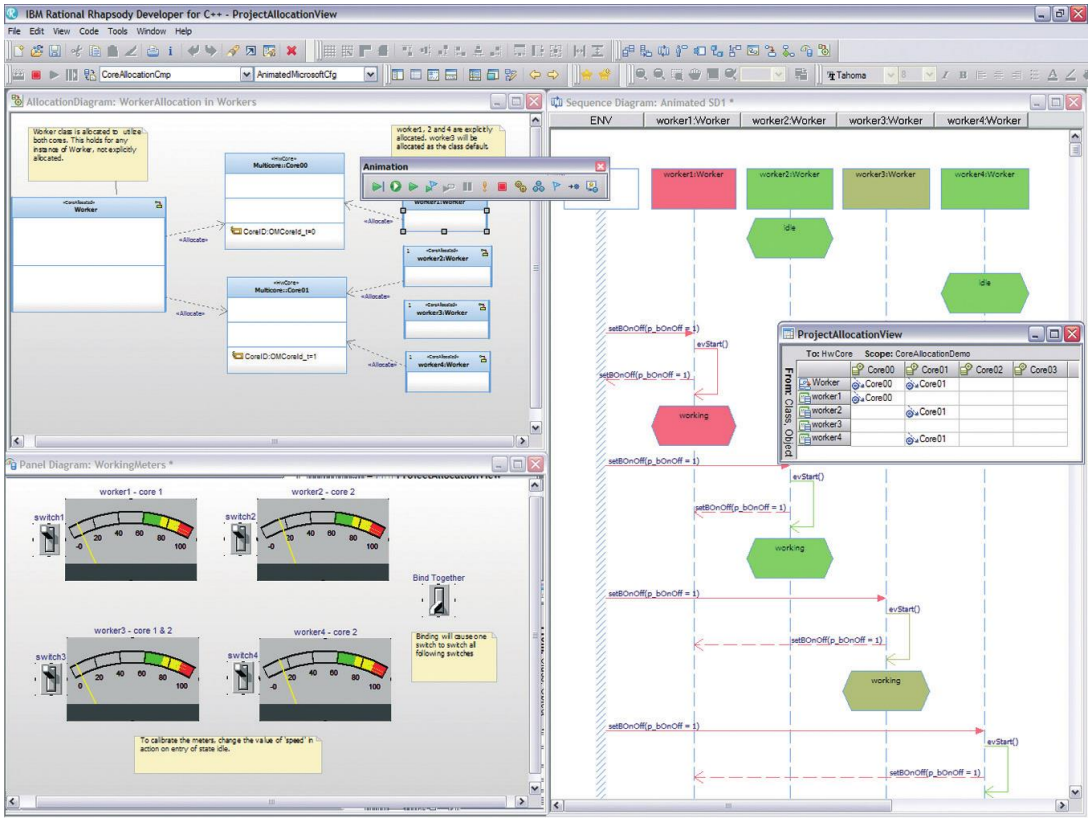

Watch bracelet manufacturing * - ArchiMetric
Project
UeXceler
Diagram
View
Team
Tools
Modeling
Window
Help
New
Undo
Redo
Copy
Paste
Select All
Format
Format Copier
Search
Group
Ungroup
Alignment
Distribution
Layout
Link
Jump
Bookmarks
Watch bracelet manufacturing
<Unspecified>
Customer
Design Dept.
Manufacturing Dept.
Delivery Dept.
Hello World Company
Issue Request
2D Conceptual Drawings
Review Prototype
Production Plan
Negotiation Proceed?
No
Yes
Accept Prototype?
No
Yes
Request Change
Place Order
Verify Product
Develop 3D Sketch
Estimate Cost
Prepare Prototype
Enter Negotiation
Issue Production Request
Prepare Invoice
Commence Production
Deliver Products

Tools

IBM Rational Rhapsody

- Description:
  - A proven solution for modeling and design activities.
- Features:
  - Visual software development environment, Collaborative development, Model-based testing, Management and traceability for integrated requirements
  - 90 day trial or Academic license
- Support:
  - UML
  - SysML
  - AUTOSAR
  - DoDAF
  - MODAF
- Homepage:
  - http://www-03.ibm.com/software/products/en/ratirhapfami

Rational® software



State-of-the-practice

What industry needs from architectural languages?

- 48 practitioners
- Use of ADLs:
  - 86% use UML or an UML profile,
  - 9% use ad hoc or in-house languages (e.g., AADL, ArchiMate)
  - 5% do not use any ADL
- Needs of ADLs:
  - Design (~66%), communication support (~36%), and analysis support (~30%)
  - Code generation and deployment support (~12% percent) and development process and methods support (~18%)
- Limitations of ADLs:
  - Insufficient expressiveness for non-functional properties (~37%)
  - Insufficient communication support for non-architects (~25%)
  - Lack of formality (~18%)

Source: Malavolta, I.; Lago, P.; Muccini, H.; Pelliccione, P. and Tang, A. What Industry Needs from Architectural Languages: A Survey IEEE Transactions on Software Engineering, 2013, v. 39, n. 6, 869-891.

State-of-the-practice

- Formalizing software architecture descriptions
  - Models must be scalable
  - Multiple formal methods must be supported
    - using multiple ADLs to model a single system
    - formalizing different aspects of a system in a single ADL
  - Incremental formalization must be supported
    - how do you formalize in the face of incompleteness?
    - **Formalize only and exactly as much as necessary**
  - Analysis results must be transferable to design and implementation
    - what good is deadlock detection at architecture alone?

Source: http://sunset.usc.edu/classes/cs599_2000/September7b.pdf


Future Directions

Future Directions

- Additional perspectives for describing software architectures
  - Runtime
  - Dynamic
  - Mobile
- Language features
  - Support multiple views
  - Customizations
  - Programming facilities
- Tools
  - Automated analysis
  - Architecture-centric development
  - Large-view management
  - Collaboration
  - Versioning
  - Knowledge management

Source: Lago, P.; Malavolta, I.; Muccini, H.; Pelliccione, P. and Tang, A. The road ahead for architectural languages. IEEE Software, 2014, 32, 98-105.

Bibliography

- ISO/IEC/IEEE 42010:2010 International Standard for Systems and Software Engineering -- Architectural description
- Malavolta, I.; Lago, P.; Muccini, H.; Pelliccione, P. and Tang, A. What Industry Needs from Architectural Languages: A Survey IEEE Transactions on Software Engineering, 2013, v. 39, n. 6, 869-891.
- Lago, P.; Malavolta, I.; Muccini, H.; Pelliccione, P. and Tang, A. The road ahead for architectural languages. IEEE Software, 2014, 32, 98-105.
- Medvidovic, N. and Taylor, R. N. A classification and comparison framework for software architecture description languages. In: IEEE Transactions on Software Engineering, 2000, v. 26, n.1, 70-93.
- Oquendo, F. pi-ADL: An Architecture Description Language based on the Higher Order Typed pi-Calculus for Specifying Dynamic and Mobile Software Architectures. In: ACM Software Engineering Notes, 2004, v. 29, n.3, 15-28.
- Clements, P.; Bachmann, F.; Bass, L.; Garlan, D.; Ivers, J.; Little, R.; Merson, P.; Nord, R.; and Stafford, J. Documenting Software Architectures: Views and Beyond. Addison-Wesley, 2011.
- Shaw, M. and Garlan, D. Characteristics of Higher-Level Languages for Software Architecture. Carnegie Mellon University, 1994. http://www.sei.cmu.edu/reports/94tr023.pdf