Model-driven Deployment & Configuration of Component-based Systems

Krishnakumar Balasubramanian, Boris Kolpackov, Tao Lu, Aniruddha Gokhale & Douglas C. Schmidt

Vanderbilt University

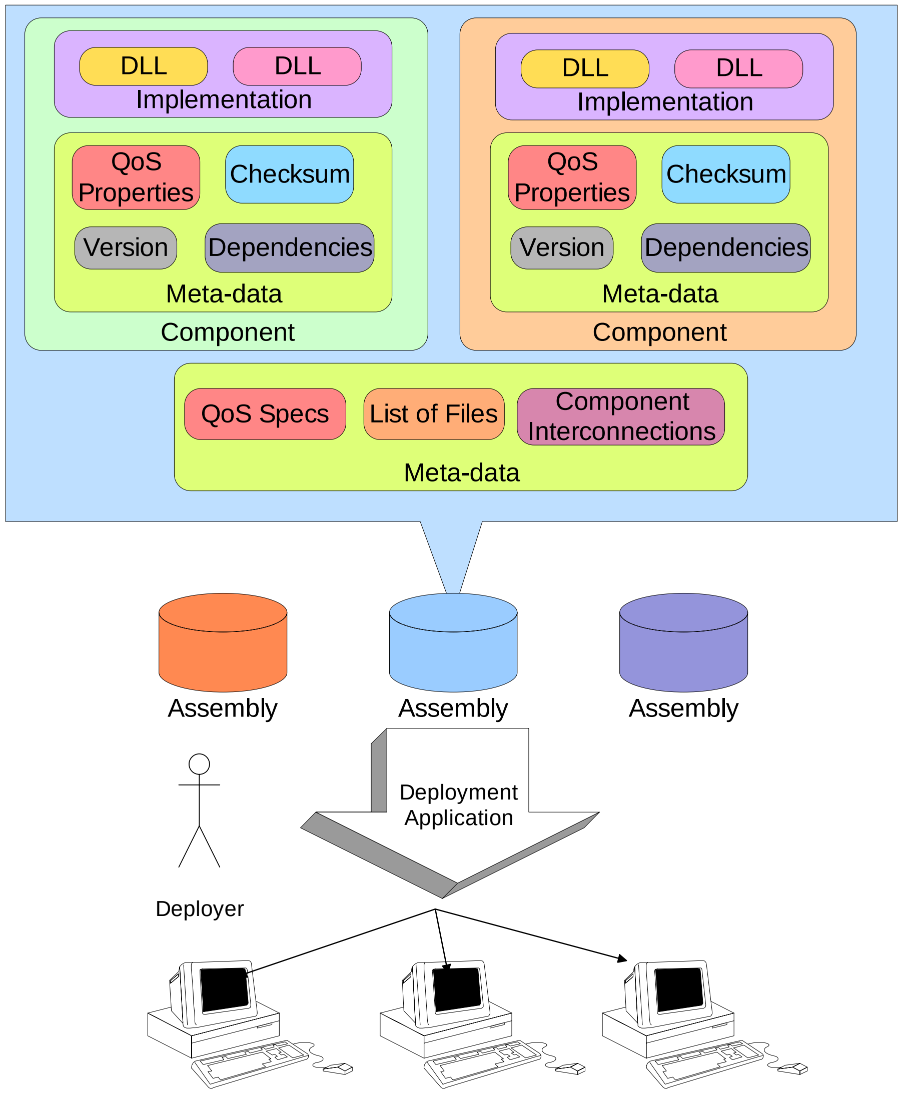
July 7, 2004

Overview

- Deployment & Configuration of systems
  - Introduction
  - Challenges
- Platform Independent Component Modeling Language (PICML)
  - How PICML addresses the challenges?
- Deployment And Configuration Engine (DAnCE)
- XML Schema Compiler (XSC)

Deployment & Configuration (D&C) Spec

- Specification defines deployment of component-based applications
- Intended to replace *Packaging & Deployment* chapter of CCM specification
- Meta-information is captured using XML descriptors
- Platform Independent Model (PIM)
- Defined in two dimensions
  - Data models vs. management (run-time) models
  - Component software vs. target vs. execution



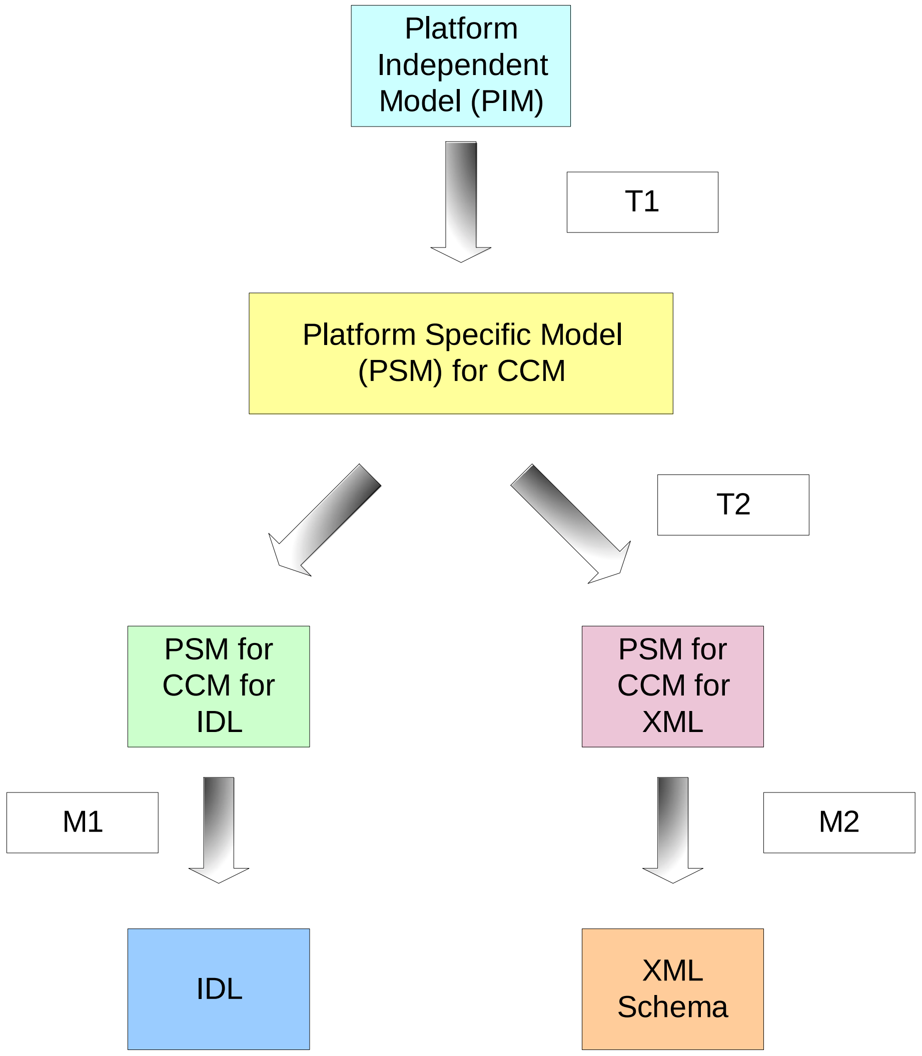
Platform-independent Model (PIM) Dimensions

- Modeling view-points
  - Conceptual, logical, & physical view-point
- Platform-independent model
  - Conceptual & logical viewpoint of deployment & configuration
- Defined in two-dimensions

PIM	Data Model	Run-time Model
Component Software	Meta-data to describe component based applications and their requirements	Interfaces to browse, store and retrieve such meta-data
Target	Meta-data to describe heterogeneous distributed systems & their capabilities	Interfaces to collect & retrieve such meta-data and commit resources
Execution	Meta-data to describe a specific deployment of an application into a distributed system	Prepare environment, Execute on target to Deployment plan, manage lifecycle

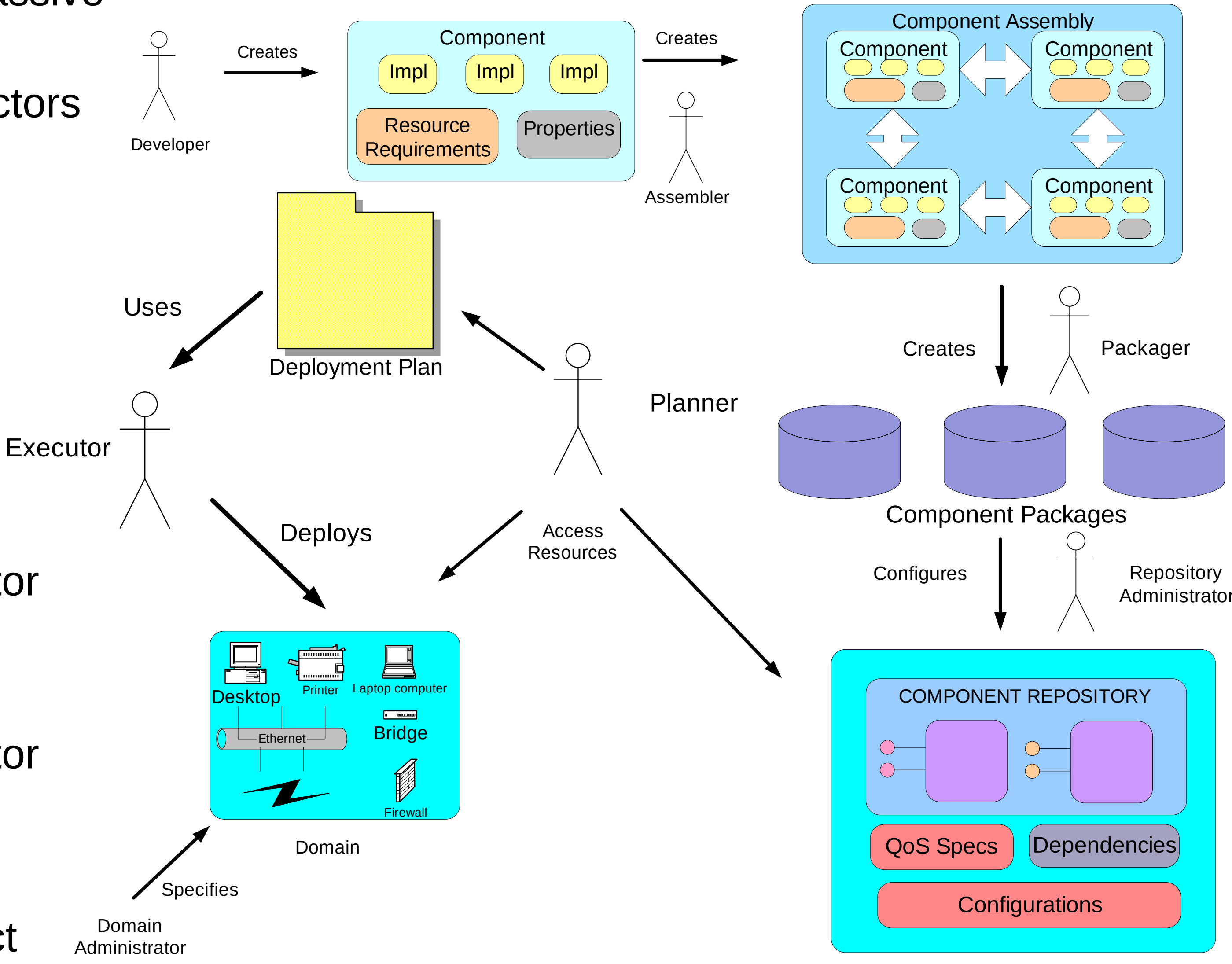
PIM Mapping to CCM

- Physical viewpoint
  - Mapping from PIM to platform specific model (PSM) for CCM
- Set of transformations
  - T1 → PIM to PSM for CCM
  - T2 → PSM to
    - PSM for IDL
    - PSM for XML
- Set of mapping rules
  - M1 → PSM to IDL
  - M2 → PSM to XML schema


Platform Independent Model (PIM)
T1
Platform Specific Model (PSM) for CCM
T2
PSM for CCM for IDL
PSM for CCM for XML
M1
M2
IDL
XML Schema

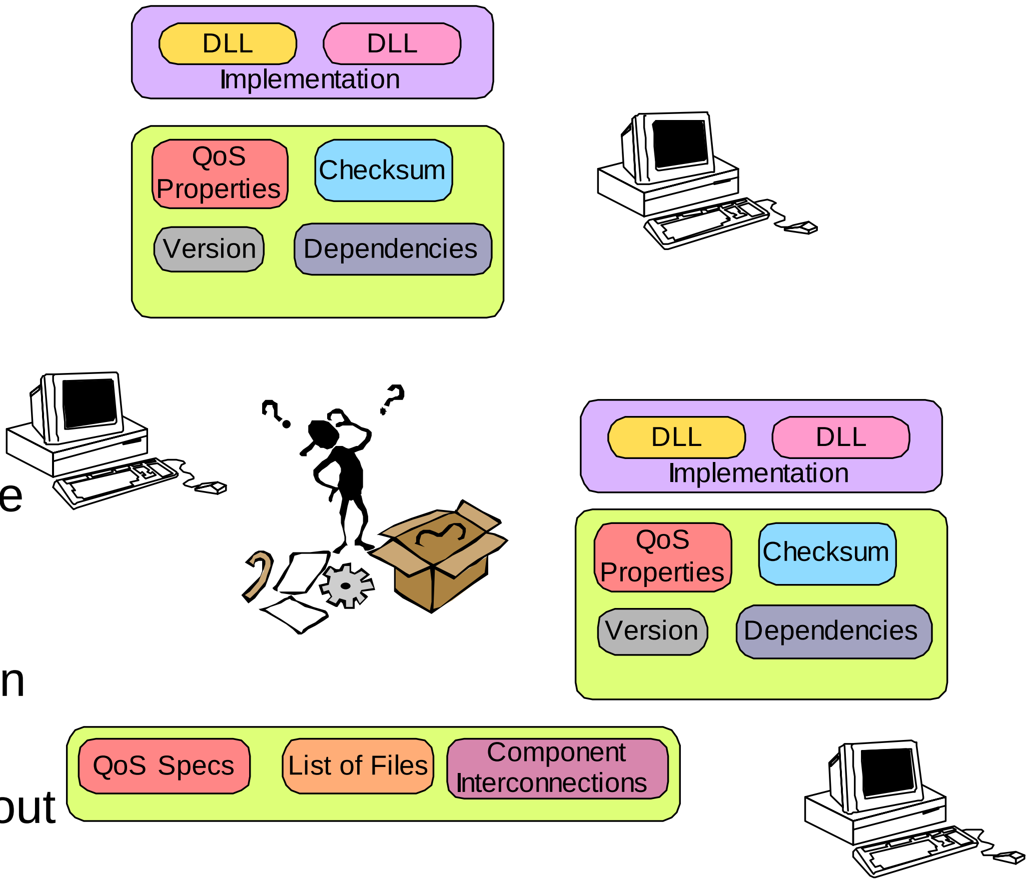
D&C Activities

- Descriptors are passive entities
- Manipulated by Actors
- Different Stages
  - *Development*
    - Developer
    - Assembler
    - Packager
  - *Target*
    - Domain Administrator
  - *Deployment*
    - Repository Administrator
    - Planner
    - Executor
- Actors are abstract



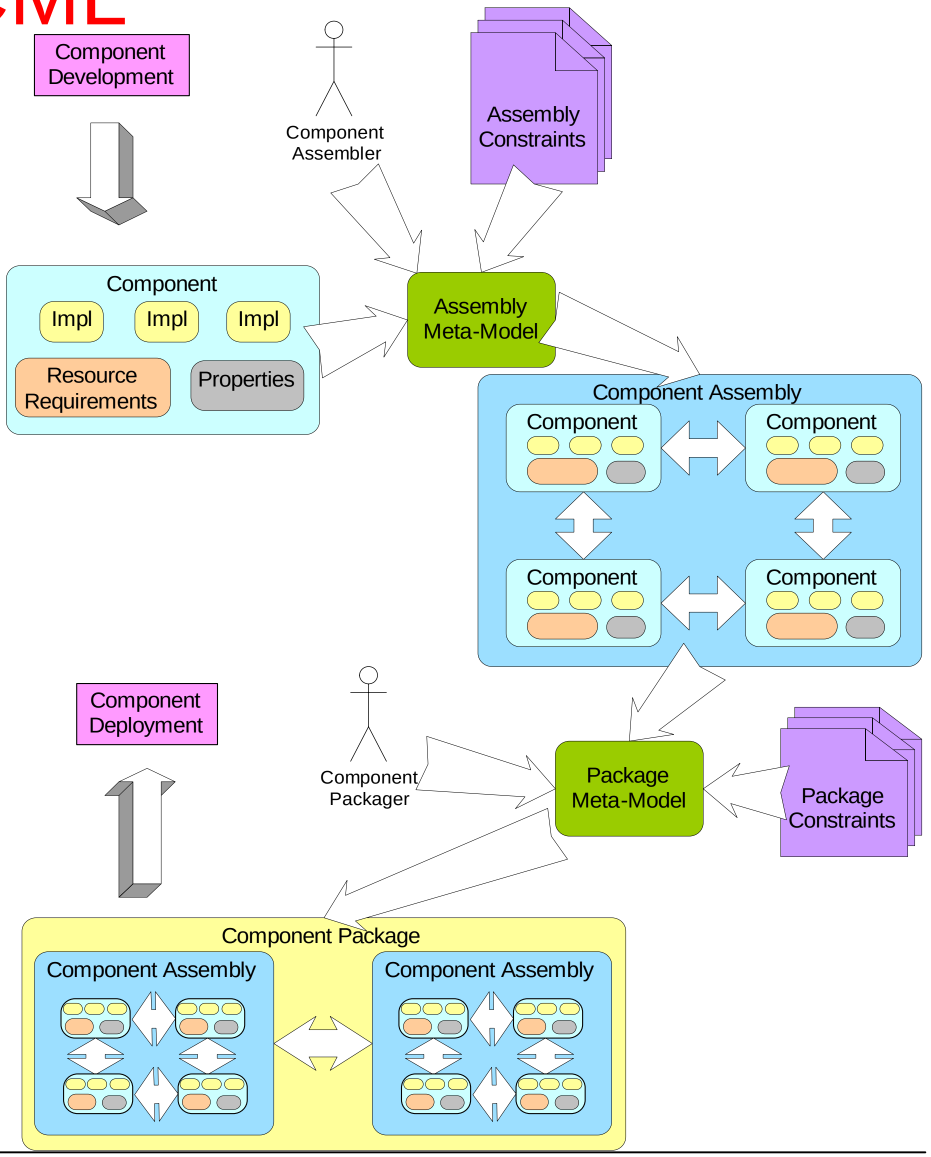
Challenges in Deployment & Configuration

- Context
  - Configuring & Deploying component-based applications using XML meta-data
- Problem
  - Meta-data split across multiple XML descriptors
  - Inter-dependencies between descriptors
  - XML is error-prone to read/write manually
  - No guarantees about semantic validity (only syntactic validation possible)
  - If meta-data is wrong, what about my data?

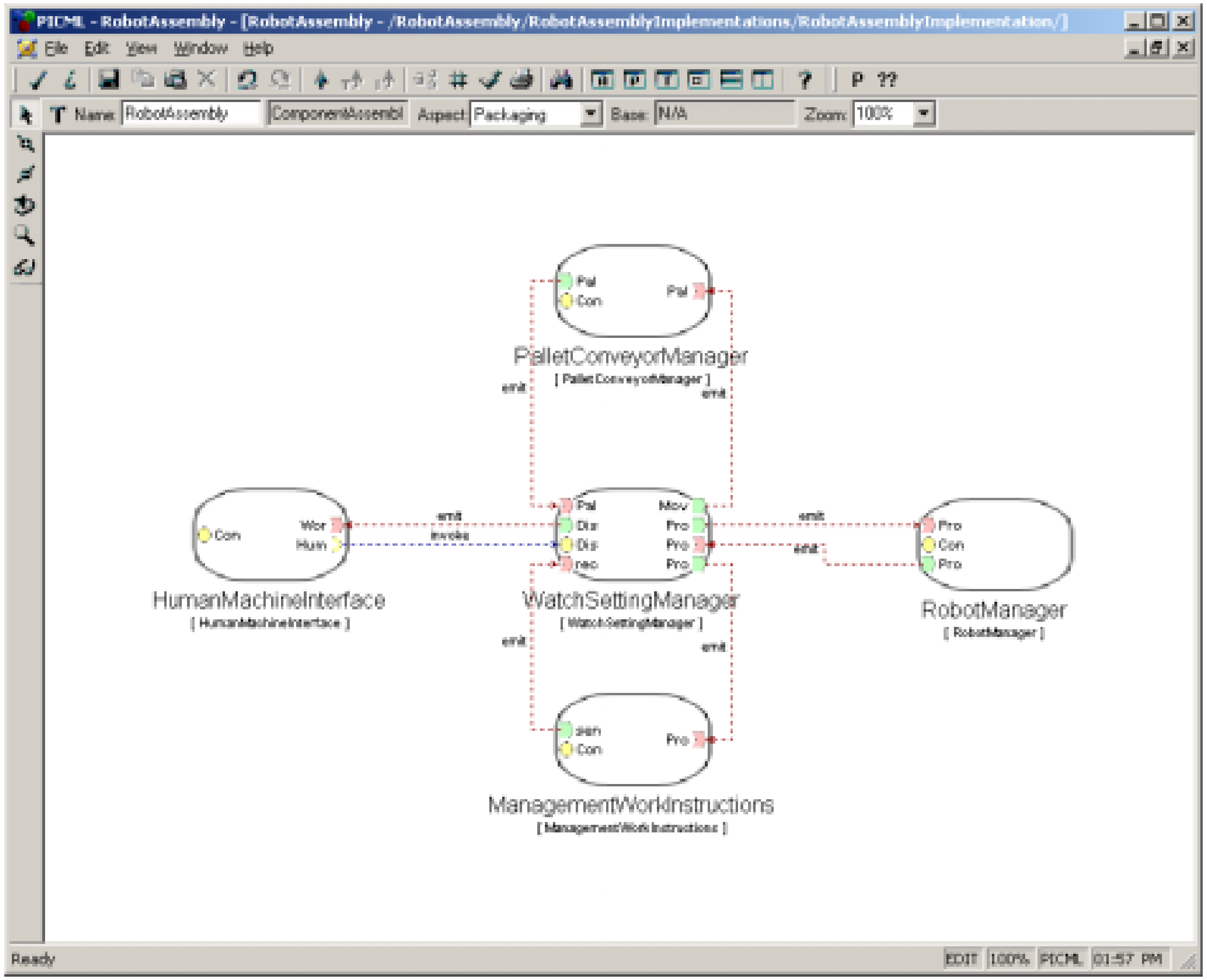


PICML

- Solution
  - Platform Independent Component Modeling Language (PICML)
    - Modeling paradigm developed using Generic Modeling Environment (GME)
  - Capture dependencies visually
  - Define semantic constraints using Object Constraint Language (OCL)
  - Generate domain specific meta-data from models
  - “Correct-by-construction”



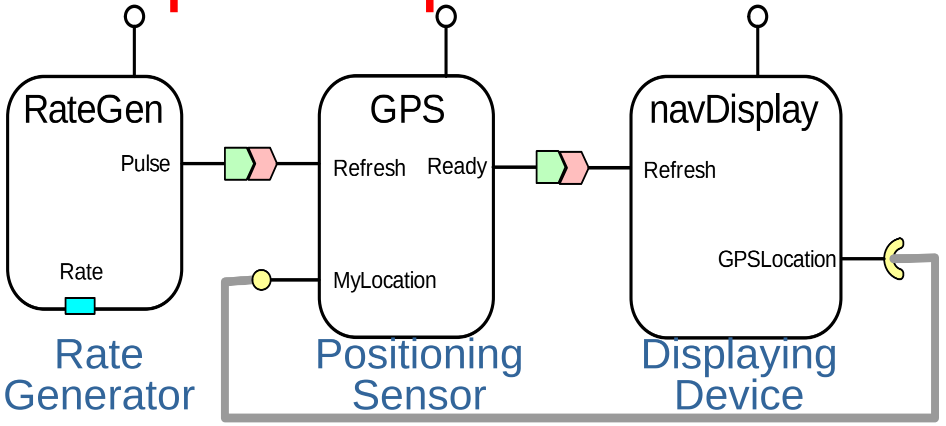
RobotAssembly Model in PICML


PICML - RobotAssembly - [RobotAssembly - /RobotAssembly/RobotAssemblyImplementations/RobotAssemblyImplementation/]
File Edit View Window Help
Name: RobotAssembly
ComponentAssembl
Aspect: Packaging
Base: N/A
Zoom: 100%
PalletConveyorManager
[PalletConveyorManager]
WatchSettingManager
[WatchSettingManager]
HumanMachineInterface
[HumanMachineInterface]
RobotManager
[RobotManager]
ManagementWorkInstructions
[ManagementWorkInstructions]
emit
invoke
Ready
EDIT 100% PICML 01:57 PM

Types of PICML generated meta-data

- **Component Interface Descriptor (.ccd)**
  - Describes the interface, ports, properties of a single component
- **Implementation Artifact Descriptor (.iad)**
  - Describes the implementation artifacts (e.g., DLLs, OS, etc.) of a single component
- **Component Package Descriptor (.cpd)**
  - Describes multiple alternative implementations of a single component
- **Package Configuration Descriptor (.pcd)**
  - Describes a specific configuration of a component package
- **Component Implementation Descriptor (.cid)**
  - Describes a specific implementation of a component interface; contains inter-connection information
- **Component Deployment Plan (.cdp)**
  - Plan which guides the actual deployment
- **Component Domain Descriptor (.cdp)**
  - Describes the target domain of deployment
- **Component Packages (.cpk)**
  - Grouping of all of the above

Example output from PICML



```
<!--Component Implementation Descriptor(.cid) associates components with impl. artifacts-->
<Deployment:ComponentImplementationDescription>
  <label>GPS Implementation</label>
  <UUID>154cf3cd-1770-4e92-b19b-8c2c921fea38</UUID>
  <implements href="GPS.ccd"/>
  <monolithicImpl>
     <primaryArtifact>
       <name>GPS Implementation artifacts</name>
       <referencedArtifact href="GPS.iad"/>
     </primaryArtifact>
  </monolithicImpl>
</Deployment:ComponentImplementationDescription>
```

Deployment And Configuration Engine (DAnCE)

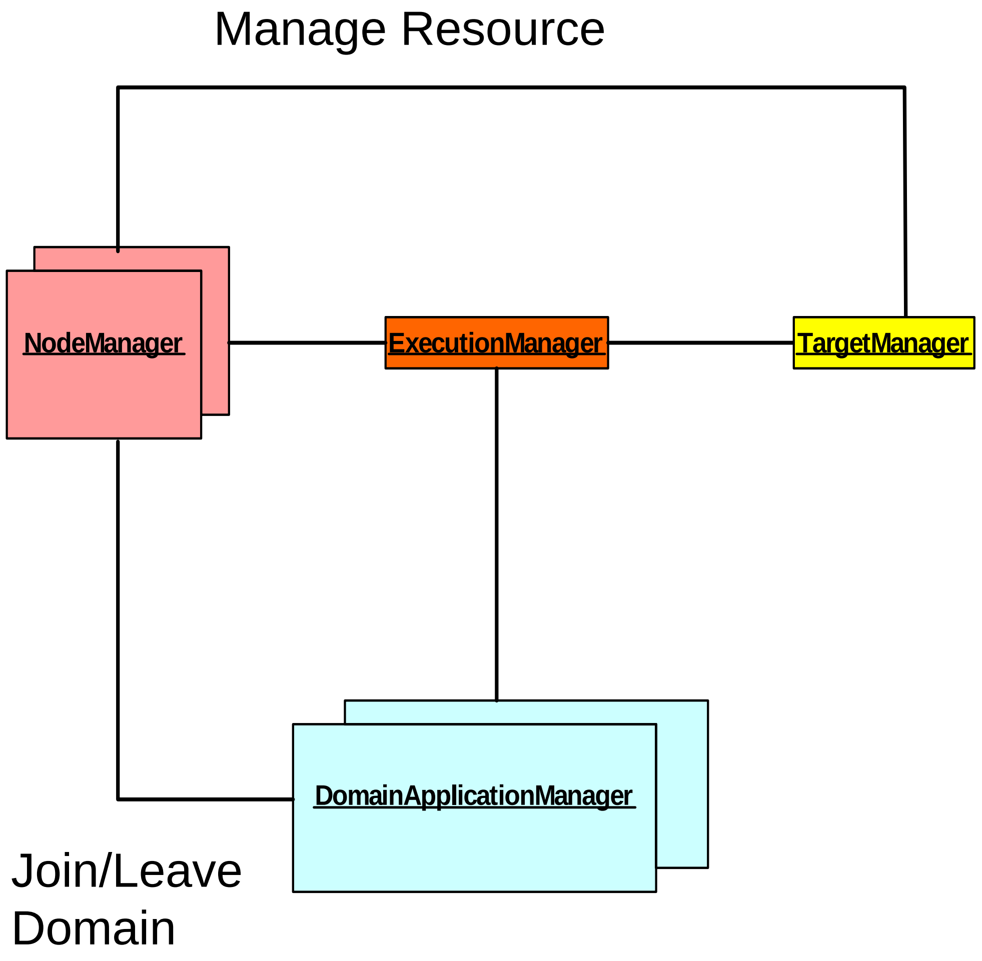
- Gather resource information.
- Local—Global resource management.
  - Commit resource
  - Verify available resource.

- Application configuration basing on the configuration generated by the PICML tool set.
  - Component configuration
  - Connection configuration

Parallel

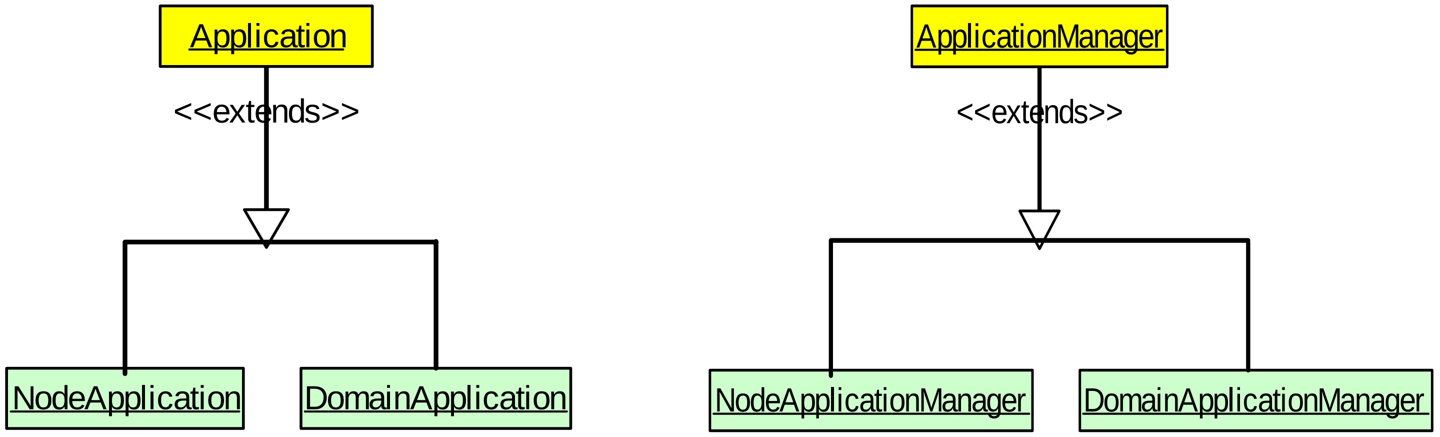
- Deploy and assemble the application.
  - Component deployment
  - Connection

DAnCE Resource Management



- Execution Manager creates DomainApplicationManagers
- NodeManager joins a Domain
- NodeManager commits resource to the Target manager for a application.
- Target manager manages global resource.
- Resource types are undefined.

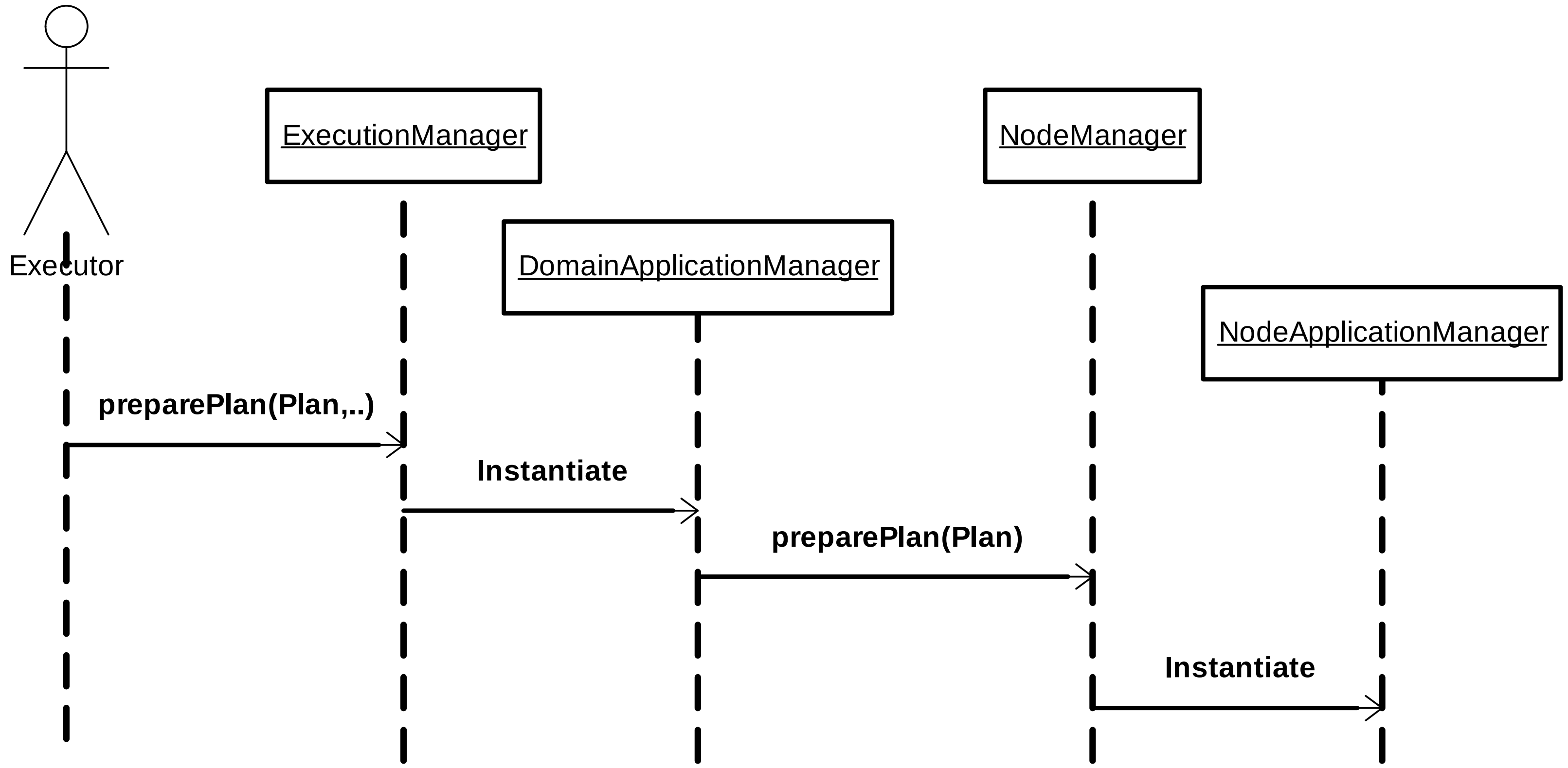
Node and Domain



- Domain* focuses on the global issues.
- Node* focuses on the local issues.

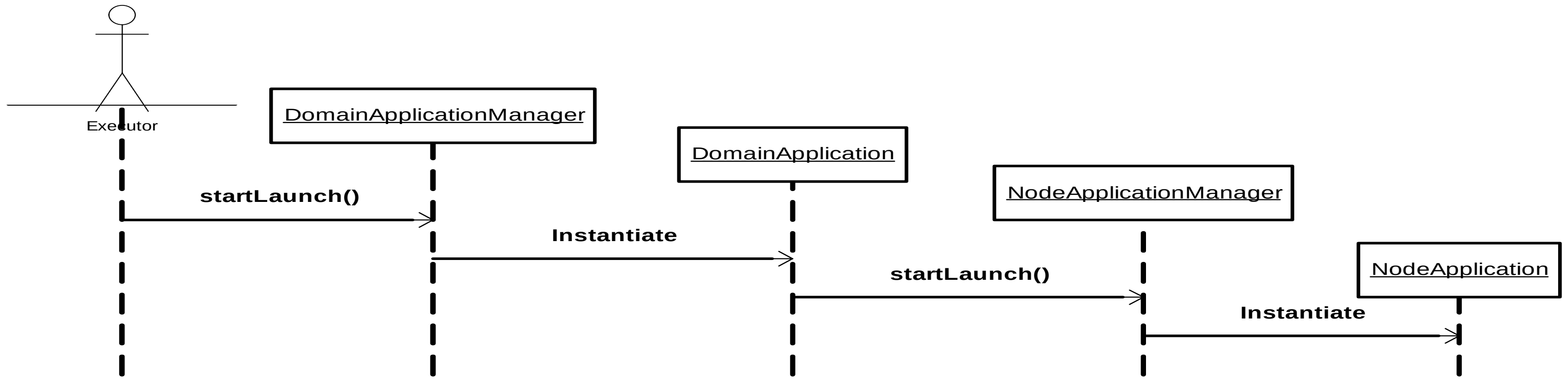
- ApplicationManager:
  - startLaunch() & destroyApplication()
- Application:
  - finishLaunch() & start()

Prepare Plan

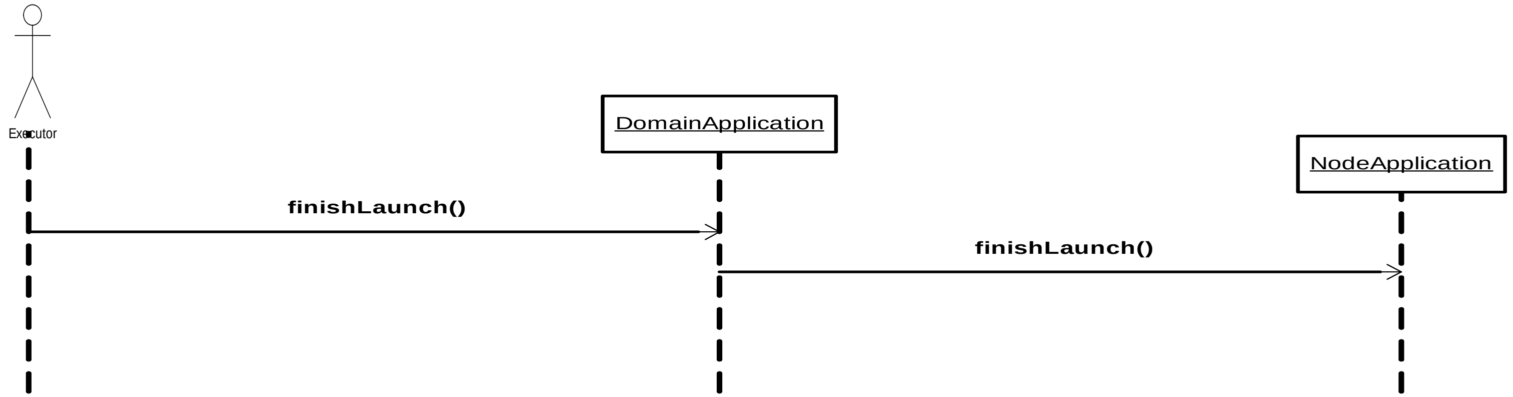


On this stage: A plan is dissembled and distributed to local targets.

Start Launch and Finish Launch



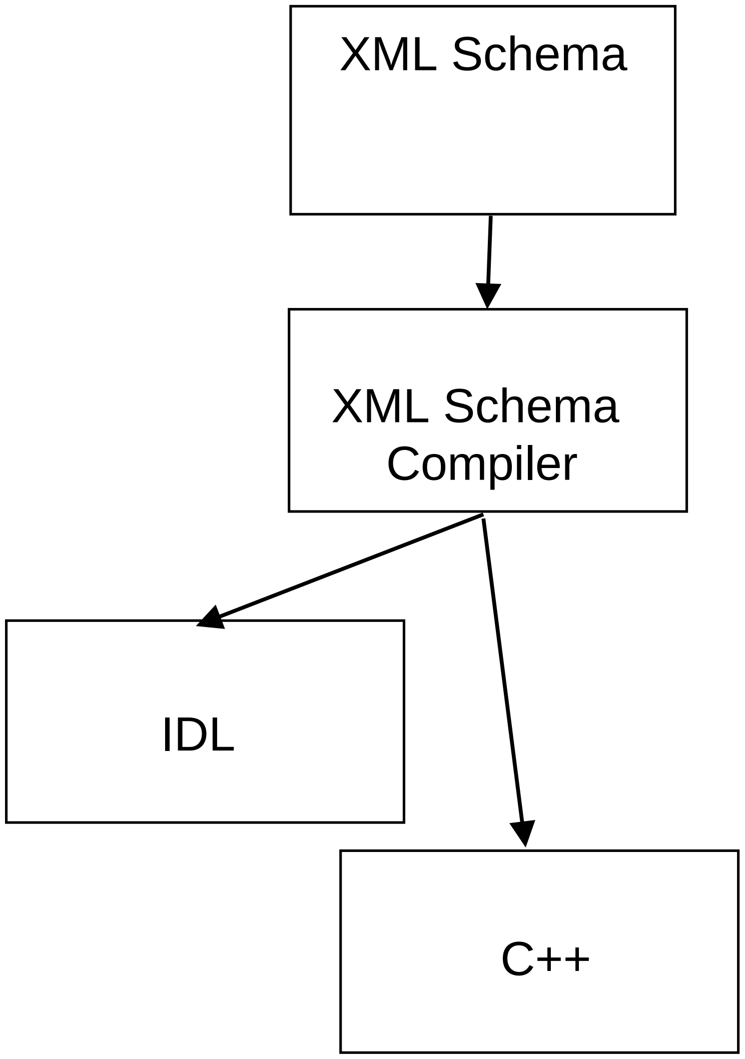
startLaunch: Components & Homes are created and configured.



finishLaunch: Connections are made.

XML Schema Compiler (XSC)

- Context
  - Increasing use of XML vocabularies as a data exchange format
- Problem
  - Standard XML APIs (SAX, DOM) are too generic and typeless
  - Results in a hard to implement/use/maintain in-memory representations.
- Solution
  - XML Schema Compiler
  - Generates statically-typed in-memory representation from schema
  - Parser and traversal mechanism for C++ or IDL.



Concluding Remarks

- Model-driven Deployment & Configuration
  - PICML
    - Models Component-based systems
    - Improves design-time validation of systems
    - Generates component meta-data
  - XSC
    - Relieves XML parsing related activities from programmers
  - DAnCE
    - Deploys component-based systems
    - Focus of future activities
- All tools available from
  - http://cvs.doc.wustl.edu (DAnCE, XSC)
  - http://cvs.dre.vanderbilt.edu (CoSMIC)