

ADL Independent Architectural Representation in XML

**Ebru Dincel
Roshanak Roshandel
Nenad Medvidovic**

University of Southern California

May 2000

**All documentation and tools developed as part of this report can be found online at
<http://scf.usc.edu/~edincel>**

Table of Contents

1	Introduction	5	
1.1	About XML		5
1.2	Related Work		5
2	C2	5	
2.1	Why C2?		6
2.2	Mapping C2 to xADL		6
2.3	Extensions		6
3	Acme	7	
3.1	Why Acme?		7
3.2	ADML		8
3.3	Mapping Acme into xADL		8
4	SADL	9	
4.1	Why SADL?		10
4.2	Mapping SADL to xADL		10
5	Darwin	11	
5.1	Why Darwin?		11
5.2	Mapping Darwin into xADL		12
6	Discussion	13	
7	Acknowledgements	15	
8	References	15	
9	Appendix A	16	
A-1	DTD Specification		16
A-2	Modeling Specification		18
A-3	C2 Constraint Checker (Perl)		32

1 Introduction

1.1 About XML

The Extensible Markup Language, XML [1], is a proper subset of the Standard Generalized Markup Language, or SGML, which has been accepted as a standard by the World Wide Web Consortium (W3C) for structuring electronic documents.

However, the first and still the most widely used markup language for the web is none other than HTML, or Hypertext Markup Language. HTML's goals are achieved by using pre-defined tags to add *meaning* to data. Taking this a step further, XML adds the ability to add the concept of *relationships* between tags, which helps in capturing the semantics of a particular domain.

Such relationships are quantified in XML using the notion of Document Type Definition or DTD. DTD is a built-in feature in XML, and is used to represent metadata. This metadata describes the rules governing the relationships between elements in the XML file, or a set of constraints regarding the nesting of tags or their order. With DTD, we can specify whether tags are required or optional, and define the valid tags allowed for the particular domain of interest.

Architecture Description Languages (ADLs) form a class of new and upcoming tools for designing software systems. Having an XML based representation for several ADLs could help extend their use in industry. Furthermore it would be interesting to look at several ADLs to find out their similarities and differences, and try to come up with an XML schema to describe them.

For the purposes of this project, we experimented with C2, Acme, Darwin and SADL for to discover how an ADL-independent architectural representation might be accomplished in XML. Our updated DTDs, which are the result of this work, are outlined in the Appendix.

1.2 Related Work

Previous work in this area includes work on developing an XML schema for Acme, as well as a simple prototype for creating the XML representation for architecture described in Acme [2]. A more architecture-neutral approach is described in [3]. There, the initial step involved creating an ontology for describing a family of ADLs, xADL, and then representing architecture specific details in an appropriate namespace, which in the case of this work was C2. An example of this would be tags prefixed by xC2:y for a specific element y.

The approach taken here has been to extend the latter work, since it was more suitable for our purposes than the others, and provided a better and firmer support for our work.

2 C2

Our work on this ADL is based on the DTD provided to us by the University of California, Irvine [3]. They have adopted XML as a key technology for enabling architecture centric tool integration in the

ArchStudio 2.0 IDE. By extracting some of the most common abstractions and their relations into a top-level xADL namespace (core), it was possible to separately represent data specific to the C2 architectural style in xC2 namespace.

2.1 Why C2?

We chose C2 since we were more familiar with the architecture and we had reliable resources to obtain additional information. Furthermore, there was similar work developed by researchers who happened to be very familiar with this style as well [3].

2.2 Mapping C2 to xADL

Below is a very high level outline of the xADL. For more details, refer to section A-1 in the Appendix.

xADL

- Architecture
 - Links
- Component
 - Supports
- Component Type
 - Interface
 - Parameter
- Connector
 - Supports
- Connector Type
 - Interface
 - Parameter

C2 was very nicely integrated with this framework. Essentially, it supported main concepts such as *Components* and *Connectors*. The *Topology* is defined by *Links* that specify what is at the top and what is at the bottom. The *Interface* is achieved by mapping the *Method* to xC2:Operation and *Parameter* is mapped to xC2:Variable.

2.3 Extensions

However, the original DTD did not have the full mapping from C2 to xADL. Therefore, we added subtyping relationships, extended port representations, and expressions. We also added details to the core definitions to support more of the functionality, for both C2 and other ADLs we covered. Figure 1 shows the planner architecture in C2. The corresponding architecture modeled in XML is outlined in Section A-2.1.

Another issue that had to be dealt with was the concept of constraints in C2, such as the restriction on component-to-component links, or number of links associated with a component. Although XML is powerful as far as structural issues are concerned, when it comes to semantics, checking constraints or consistency, we realized that XML was not sufficient. Consequently we developed a constraint-checker tool in Perl (see section A-3 in the Appendix), to provide us with those capabilities. In an ideal case, these tools should be part of an integrated environment, such as

ArchStudio. However, due to specific circumstances (described below), we could not take advantage of such an environment.

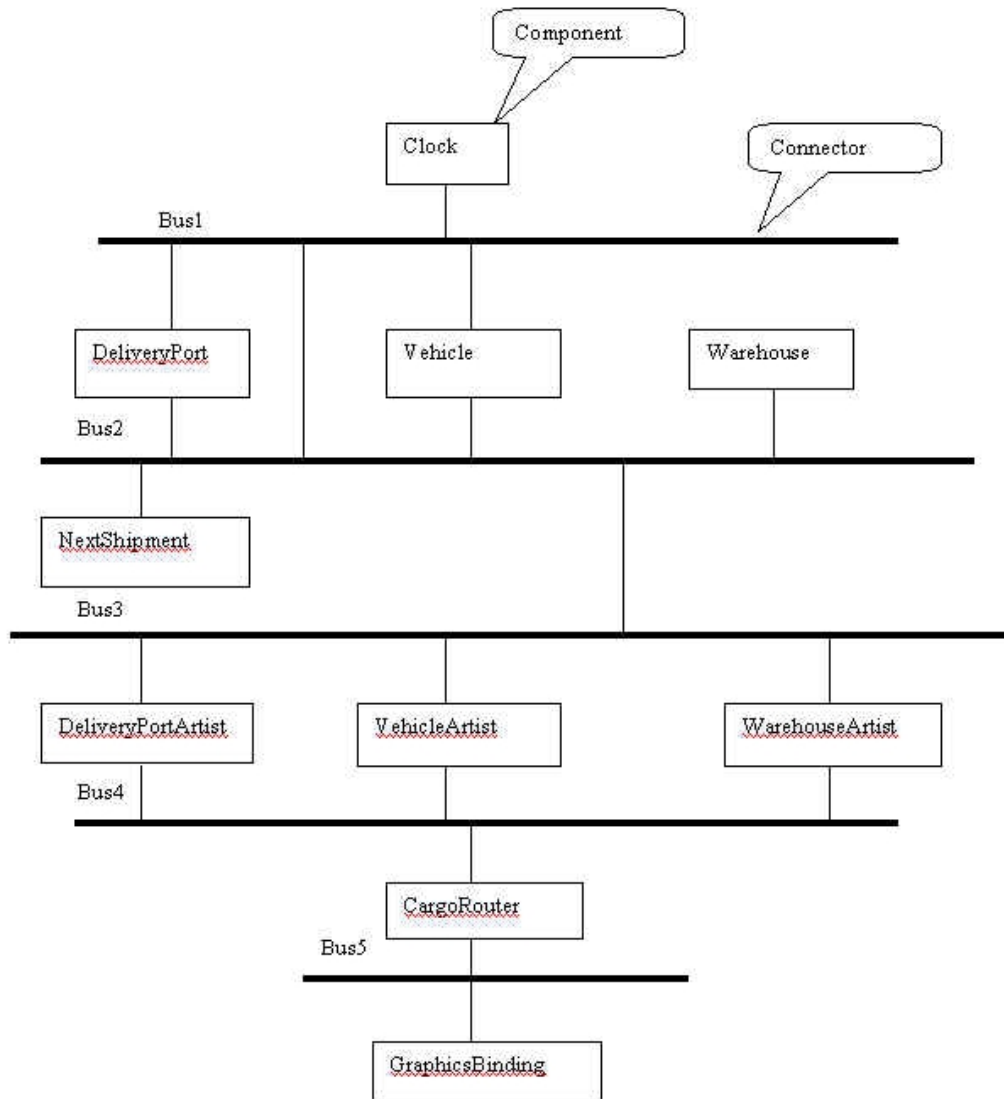


Figure 1: Planner Architecture in C2.

3 Acme

3.1 Why Acme?

There are several ADLs in the software architecture research community, each of which is more specialized toward a certain domain or is capable of certain utilities that others may lack. Each ADL supporting tool serves certain purposes. However Acme claims to provide an interchange

between wide varieties of architecture tools. Acme also can serve as an ADL by itself by providing a set of language constructs for describing architectural structure, types, architectural styles and properties of the architectural elements. All of the above makes Acme a good candidate for our work, since we are looking at several ADLs for analysis.

3.2 ADML

There has been another effort to create a DTD and XML representation for Acme by the Microelectronics and Computer Technology Corporation (MCC), known as ADML [2]. While it is a very comprehensive work, we found it too detailed in trying to explain the language features. ADML is describing the grammar of the language comprehensively. However we were interested mostly in the intersection of Acme with other ADLs, as well as Acme's conceptual feature, more than the syntax and grammar of the language.

3.3 Mapping Acme into xADL

Below you will find a description of relevant Acme features of interest to us here.

Acme

- System
 - Component
 - Port
 - Property
 - Representation
 - Connector
 - Role
 - property
 - Representation
 - System
 - Binding
 - Attachment
 - Links

There are certain features in Acme, which can be mapped almost completely into xADL features. By doing so however, we inevitably lost some of the Acme specific terminology. However, it is our belief that since the mapping does not affect any language features, it does not create major concerns for us.

Some Acme concepts can be directly mapped to xADL by adding optional items to them. However some others did not have any correspondence in other ADLs, and so we had to model them independently under the xAcme namespace.

Acme	xADL
Design	xADL
System	Architecture
Links	Links
Component	Component
Connector	Connector
Property	Property

ComponentType	xAcme:ComponentType
ConnectorType	xAcme:ConnectorType
Port	Port
Role	xAcme:Role
PortType	xAcme:PortType
RoleType	xAcme:RoleType
PropertyType	xAcme:PropertyType
Family	xAcme:Family
Representation	xAcme:representation

Table 1: Mapping of Acme concepts into xADL.

We need to emphasize that our goal has not been to write a complete XML representation for Acme, as this has been already done extensively in ADML. Rather we tried to find Acme constructs that are shared or related to other ADLs, and that can be used in conjunction with other ADLs to represent a software architecture.

Figure 2 shows an example of a client-server architecture. In sections A-2.3 and A-2.4 in the Appendix you can find two XML representations for this example. The former is a simple architecture while the latter supports the concept of *Properties* in Acme. One major problem was the lack of appropriate resources to find a more detailed architecture described with Acme. Unlike the C2 case where we had quite a few examples already described in C2, the number and complexity of Acme examples were very much limited.

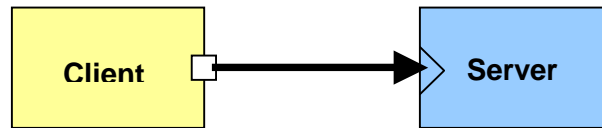


Figure 2: Client-Server architecture in Acme.

4 SADL

SADL is a language to specify both the structure and the semantics of an architecture, although its main focus has been on the former. SADL can be used to specify specific architectures (in which every architectural element has a specific name), generic architectures, mappings (relating two architectures by a syntactic interpretation mapping), architectural styles, and architecture refinement patterns [4]. Below is the representation of the architectural elements we mapped to xADL.

4.1 Why SADL?

We chose SADL because we were able to readily find reference manuals and information on it. SADL also accepts most of the standardized architectural elements, which is a desirable feature.

4.2 Mapping SADL to xADL

SADL

- Architecture

 - Component

 - Port

 - Interface

 - Connector

 - Configuration

 - Connections

 - Constraints

Table 2 shows the mapping of the above elements to appropriate xADL elements.

SADL	xADL
Importing	xSADL:Importing
Exporting	xSADL:Exporting
Link	Link
Component	Component
Connector	Connector
Port	Port
Constraint	xSADL:Constraint

Table 2: Mapping of SADL concepts into xADL.

The SADL constraint language is provided with formal semantics (extended first order logic). As such XML's DTD was unable to inherit all the power this extension offered. Although we did model certain primary expressible constraints such as the read/write constraint or the ordering constraint, it was not possible to cover some other constraints given XML limitations. Implementing refinement capabilities of SADL was beyond the scope of this work, due to such limitations, and because it involved constraining system behavior.

Figure 3 shows a level 2 compiler architecture in SADL. This architecture is modeled in Section A-2.2. Since this ADL is in compliance with software architecture terminology, it was not hard to map the architectural elements, interfaces, ports, etc. to the core of our DTD. However, SADL extensions could not get integrated due to XML's weakness in supporting such formalism.

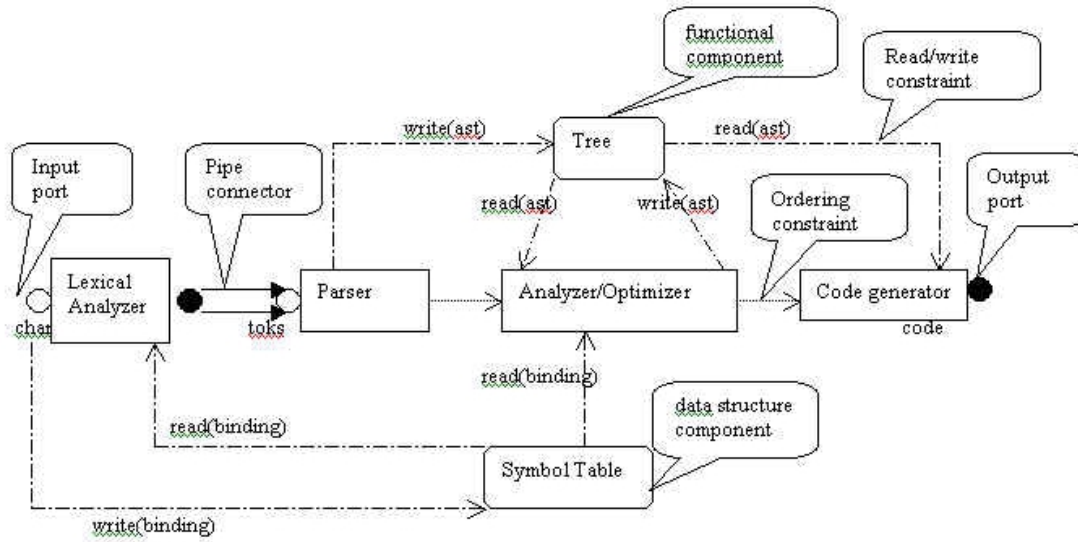


Figure 3: Level 2 Compiler Architecture in SADL

5 Darwin

Darwin is another ADL that we picked for our work here. It is used primarily in describing distributed system structures. Its operational semantics are based on π -calculus, an elementary calculus for describing and analyzing concurrent systems with evolving communication structure. Its details and formalism are beyond the scope of this discussion. Interested readers can refer to the list of references for details [5].

5.1 Why Darwin?

Software architecture and design is a dynamic process. Many of the available ADLs do not address the issue of dynamism, which makes it necessary for us to look at a system which has ways to specify this behavior. Unfortunately the available resources on Darwin are very limited and as a result we did not have access to the grammar and details of the structure of the language. As

a result we took a reverse-engineering oriented approach to include Darwin in our xADL. This study by no means is comprehensive and definitely needs more time and resources for full modeling. As a result our example architectures in Darwin are meant as proof-of-principle only, and are based on the examples presented in a couple of resources available.

5.2 Mapping Darwin into xADL

Table 3 below displays a mapping from Darwin to xADL, again with regard to the added elements and attributes specific to Darwin.

Darwin	XADL
Component	Component
ComponenetType	xDarwin:ComponentType
Provide	Port
Require	Port
Inst	xDarwin:Inst
Bind	Link

Table 3: Mapping of Darwin concepts into xADL.

Figures 4 and 5 show two simple examples of architectures in Darwin. The corresponding XML representations can be found in sections A-2.5 and A-2.6 of the Appendix.

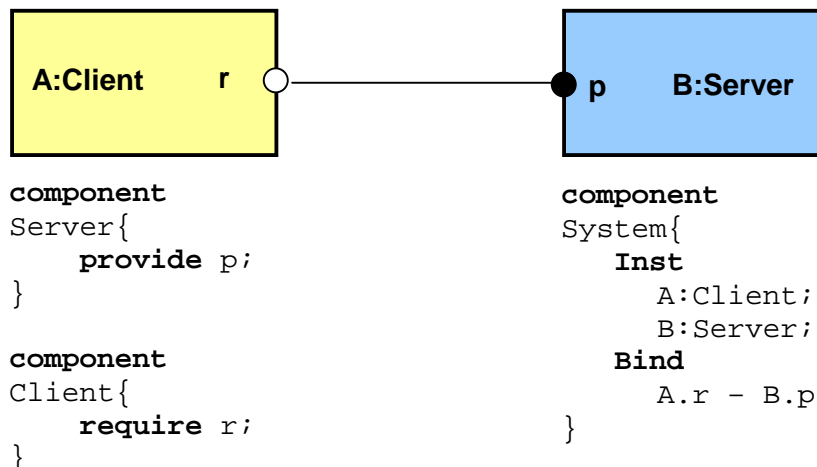


Figure 4: Client-server Architecture in Darwin.

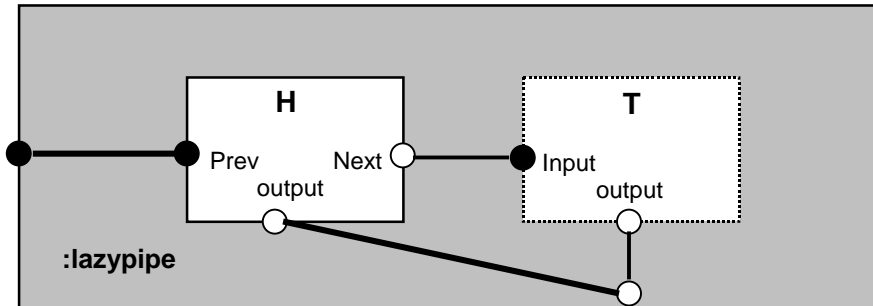


Figure 5: Lazypipe Architecture in Darwin.

6 Discussion

Prior to representing architectures in an ADL-independent manner, we looked at several ADLs including C2, SADL, Darwin and Acme; and analyzed them, both semantically and structurally. However we placed more emphasis on the semantic aspect, in order to establish a basis for interchange among different ADLs. xADL, a DTD developed by University of California, Irvine provided a good basis for this. The core of this DTD contains concepts that are well accepted and shared by most ADLs. In addition to ADLs specified above, we extended xADL to handle ADL-specific concepts through an XML feature known as namespaces. Namespaces ensure uniqueness among XML elements. These also reflect our original idea of having an ADL-independent representation in XML.

The representation could be achieved in at least two ways: The first one involves keeping the core as simple as possible and relying on the extensions to represent the language-specific features. This way someone who is familiar with a specific ADL would feel more comfortable working in this environment since the language-specific terminology is preserved. On the other hand, it might degrade into being very specialized if the namespace facility is abused. This would lead to several DTDs for each ADL.

The second approach would be to rely upon the core as the backbone of this framework so that it reflects the commonalities between the ADLs without necessarily preserving syntactic information and terminology, as well as differences that might be tailored to fit into this core. Although it might provide a better interchange capability, specific terminology is lost; assumptions about this mapping might go unstated; and a person might be clueless as to how to take advantage of this environment unless equipped with detailed documentation.

Since each of us had a different vision on how to attack this problem, we decided to go somewhere in between as a compromise, in order to balance the advantages and disadvantages of these approaches for this particular project.

We considered several possibilities for having tool support to realize our ideas. One was to extend ArchStudio 2.0, which already had some implementation for xADL. This way we could have benefited from constraint checking, and automatic XML generation from a given DTD. The main problem was that we did not have access to the integrated environment early enough to explore them. Furthermore since our work was not just C2 related, we were not sure if that would be beneficial to investigate given the time limit and the effort involved. Also, there were some online tools available to parse the DTD to ensure that it is valid, and to convert it to XML by instantiating it with specific parameters (<http://www.xmlschema.com>). AcmeStudio was another option but we had a similar access problem because of broken web links, and poorly maintained resources.

This is why we developed a simple Perl program that simply checks for certain patterns in the XML file to make sure that C2 constraints are not violated, such as multiple links from a component or component-to-component communication. Perl is flexible enough to check these constraints since the input XML files are, by definition, structured. However, developing an extensive tool to handle all ADL specific constraints needed a much more in depth understanding of all these ADL-specific semantics, which we lacked. We also think that available tool support indirectly relates to the approach we might have adopted. It is hard to isolate the concepts from each other.

XML provides a flexible and extensible framework suited to ADLs. There are conceptual similarities between most of the ADLS making a standard description possible. As XML becomes more of a widely used standard for structured information representation on the Internet, it might be beneficial to share different ADL models through this common structure. Although this standardization might have a promising future, it can only be really powerful when architecture description languages, as well as the architecture research reach a more mature level.

Our work can be compared with other languages that claim to serve a similar purpose, such as Unified Modeling Language (UML) and Acme.

Both UML and XML have extensibility mechanisms to accommodate future changes to an ADL's definition. XML supports this by new tags and attributes where UML has stereotyping extension. Analysis techniques are provided by both of them. In the case of XML, there are online tools, parsers, validation tools, and converters, whereas in UML this is somewhat internally embedded and may need to be extended further for supporting software architecture concerns. Although XML representations claim to establish traceability with XML linking facilities, there is much more to this issue on a semantic level. UML would provide a better framework as far as this issue goes. Both UML and XML provide different views to fulfill communication. However, XML's views tend to be architecture specific whereas UML's views are more audience specific. XML's richness is more related to its structural representation; however, one would also like to assess semantic mappings or formalism when trying to create an ADL-independent architectural representation.

Acme is an interchange language for ADLs that is widely known among the architecture research community. What makes our approach different from Acme is that we are not trying to convert one ADL to another, by using xADL. We just tried to find the similar aspects of some ADLs and represent them with XML. As a result, our work does not involve the complex conversion that Acme involves. Our approach is potentially more appropriate for designing subsystems of a software system in different ADLs, and possibly making them interact through our connectors which can encompass more than one ADL's requirements.

7 Acknowledgements

First of all, we would like to thank Professor David Wile, Yuzo Kanomata, and Peyman Oreizy for insightful discussions, materials they have provided, and their kindness.

8 References

- [1] R. Eckstein, *XML Pocket Reference*, O'Reilly and Associates, 1999.
- [2] C. Goyette, "XML applied to product line software development," available online at <http://www.mcc.com/projects/ssepp/papers/XmlApplied.htm>
- [3] R. Khare, M. Guntersdorfer, P. Oreizy, N. Medvidovic, and R. Taylor, "xADL: enabling architecture-centric tool integration with XML".
- [4] M. Moriconi, R.A. Riemenschneider, "Introduction to SADL 1.0: a language for specifying architecture hierarchies", Technical Report SRI-CSL-97-01.
- [5] J. Magee and J. Kramer, "Dynamic structure in software architectures", *Proc. ACM SIGSOFT*, pp. 3-13, 1996.
- [6] S. Laurent, *XML Elements of Style*, McGraw Hill, 2000.
- [7] T. Christiansen and N. Torkington, *Perl Cookbook*, O'Reilly and Associates, 1999.
- [8] D. Martin et. al., *Professional XML*, Wrox Press, 2000.
- [9] J. Robbins, N. Medvidovic, D. Redmiles, and D. Rosenblum, "Integrating architecture description languages with a standard design method" *Proc. 20th Int. Conf. on Software Engineering (ICSE'98)*, pp. 209-218, Kyoto, Japan, April 19-25, 1998.
- [10] D. Garlan, R. Monroe, and D. Wile, "Acme: an architecture description Interchange language", *Proc. CASCON 97*, pp 169-183, November 1997, available online at http://www.cs.cmu.edu/afs/cs.cmu.edu/project/able/www/paper_abstracts/acme-cascon97.html.
- [11] N. Medvidovic and R. Taylor, "A classification and comparison framework for software architecture description languages". *IEEE Transactions on Software Engineering*, to appear, 2000.
- [12] D. Garlan and Z. Wang, "A case study in software architecture Interchange", available online at http://www.cs.cmu.edu/afs/cs/project/able/www/paper_abstracts/acme-wr2rap98.html.
- [13] A. Kompanek "Modeling a system with Acme", available online at http://www.cs.cmu.edu/~acme/acme_extending_acme.html
- [14] N. Medvidovic, D. Rosenblum, and R. Taylor, "A language and environment for architecture-based software development and evolution". *Proc. 21st Int. Conf. on Software Engineering, (ICSE'99)*, pp. 44-53, Los Angeles, CA, May 16-22, 1999.

9 Appendix A

A-1 DTD Specification

```

<?xml version="1.0" encoding="US-ASCII"?>

<! ELEMENT   xADL (ComponentType | ConnectorType | Architecture
|xAcme: ComponentType | xAcme: ConnectorType | PortType | xAcme: RoleType |
xAcme: PropertyType | xAcme: Family | xDarwin: ComponentType )*>

<! ELEMENT   Architecture ( xSADL: Exporting | xSADL: Importing| Component |
Connector |Topology | Port |xAcme: Role | xAcme: Property | xAcme: Representati on)*
>
<! ATTLIST   Architecture name CDATA #REQUIRED>

<! ELEMENT   xSADL: Importing (xSADL: ImportList)+ >
<! ATTLIST   xSADL: Importing from CDATA #REQUIRED>

<! ELEMENT   xSADL: Exporting (xSADL: ExportList)+ >

<! ELEMENT   xSADL: ExportList (#PCDATA) >
<! ELEMENT   xSADL: ImportList (#PCDATA) >

<! ELEMENT   Topology (Link*, xSADL: Constraint*) >

<! ELEMENT   Link (Port*) >
<! ATTLIST   Link from CDATA #REQUIRED
              to CDATA #REQUIRED
              name CDATA #IMPLIED
              conname CDATA #IMPLIED>

<! ELEMENT   xSADL: Constraint EMPTY>
<! ATTLIST   xSADL: Constraint name CDATA #REQUIRED
                              type CDATA #REQUIRED
                              source CDATA #IMPLIED
                              destination CDATA #IMPLIED >

<! ELEMENT   ComponentType (Supports*, Method*, xC2: Subtype*, xC2: Behavior?,
Component*)>
<! ATTLIST   ComponentType name CDATA #REQUIRED
                              xC2: OS CDATA #IMPLIED
                              xC2: ImplementationModule CDATA #REQUIRED
                              xC2: Language CDATA #IMPLIED>

<! ELEMENT   ConnectorType (Supports*)>
<! ATTLIST   ConnectorType name CDATA #REQUIRED
xC2: filter (no_filtering | notification_filtering |message_filtering |
prioritized | message_sink ) 'no_filtering'
              xC2: OS CDATA #IMPLIED
              xC2: Language CDATA #IMPLIED>

```

```

<! ELEMENT      Method      (Parameter*)>
<! ATTLIST     Method      name CDATA #REQUIRED
                        xC2:direction (provide | require) #REQUIRED
                        xC2:mapToOper IDREF #IMPLIED >

<! ELEMENT      Parameter EMPTY>
<! ATTLIST     Parameter   name CDATA #IMPLIED
                        type CDATA #REQUIRED
                        role ( in | out | inout ) 'in'
                        xC2:mapToVar IDREF #IMPLIED >

<! ELEMENT      Component (Port*, Supports*, xAcme: Property*, xDarwin: Inst*,
Link*)>
<! ATTLIST     Component   name CDATA #REQUIRED
                        type CDATA #IMPLIED
                        datatype CDATA #IMPLIED>

<! ELEMENT      Connector (Port*, Supports*, xAcme: Role*, xAcme: Property*)>
<! ATTLIST     Connector   name CDATA #REQUIRED
                        type CDATA #IMPLIED
                        datatype CDATA #IMPLIED>

<! ELEMENT      Port EMPTY>
<! ATTLIST     Port        type ( in | out | top | bottom) #IMPLIED
                        datatype CDATA #IMPLIED
                        name CDATA #IMPLIED
                        property CDATA #IMPLIED>

<! ELEMENT      Supports EMPTY>
<! ATTLIST     Supports   type CDATA #REQUIRED >

<! ELEMENT      xC2: Behavi or (xC2: State, xC2: Invari ant*, xC2: Operati on*)>
<! ATTLIST     xC2: Behavi or name CDATA #IMPLIED>

<! ELEMENT      xC2: State (xC2: Vari abl e | xC2: Functi on)* >

<! ELEMENT      xC2: Subtype (#PCDATA) >
<! ATTLIST     xC2: Subtype name CDATA #REQUIRED
                        type (nam | int | beh | imp | notnam | notint | notbeh |
notimp) #REQUIRED>

<! ELEMENT      xC2: Vari abl e EMPTY>
<! ATTLIST     xC2: Vari abl e      name CDATA #REQUIRED
                        type CDATA #REQUIRED
                        uid CDATA #REQUIRED >

<! ELEMENT      xC2: Functi on EMPTY>
<! ATTLIST     xC2: Functi on      name CDATA #REQUIRED
                        from CDATA #REQUIRED
                        to CDATA #REQUIRED >

<! ELEMENT      xC2: Invari ant (xC2: Expressi on)+>

<! ELEMENT      xC2: Expressi on EMPTY>
<! ATTLIST     xC2: Expressi on op1 CDATA #REQUIRED
                        optype (equal sto | setequal sto| notequal sto| addi ti on
| setaddi ti on| subtracti on | setsubtracti on| mul ti pli cation | di vi si on |
exponenti ation | implies | equivalent | and | or | uni on | intersecti on | in |
notin | greater | less | eqgreater | eql ess | seteqgreater | seteql ess)#REQUIRED
                        op2 CDATA #REQUIRED>

```

```

<! ELEMENT      xC2: Operati on (xC2: Let*, xC2: Pre*, xC2: Post*)>
<! ATTLIST     xC2: Operati on      name CDATA #REQUIRED
              directi on (provi de | requi re ) #REQUIRED
              ui d CDATA #REQUIRED >

<! ELEMENT     xC2: Let      (xC2: Vari abl e)*>
<! ELEMENT     xC2: Pre      (xC2: Expressi on)>
<! ELEMENT     xC2: Post     (xC2: Expressi on)>

<! ELEMENT     xAcme: Rol e  EMPTY>
<! ATTLIST     xAcme: Rol e  name CDATA #REQUIRED
              property CDATA #IMPLIED >

<! ELEMENT     xAcme: Property  EMPTY>
<! ATTLIST     xAcme: Property  name CDATA #REQUIRED
              type CDATA #REQUIRED
              val ue CDATA #REQUIRED>

<! ELEMENT     xAcme: ComponentType (Port*, xAcme: Property*)>
<! ATTLIST     xAcme: ComponentType i denti fi er ID #REQUIRED>

<! ELEMENT     xAcme: ConnectorType (xAcme: Rol e*, xAcme: Property*)>
<! ATTLIST     xAcme: ConnectorType i denti fi er ID #REQUIRED>

<! ELEMENT     PortType  EMPTY>
<! ATTLIST     PortType  i denti fi er ID #REQUIRED>

<! ELEMENT     xAcme: Rol eType  EMPTY>
<! ATTLIST     xAcme: Rol eType  i denti fi er ID #REQUIRED>

<! ELEMENT     xAcme: PropertyType  EMPTY>
<! ATTLIST     xAcme: PropertyType  name CDATA #REQUIRED
              type CDATA #REQUIRED
              val ue CDATA #REQUIRED>

<! ELEMENT     xAcme: Fami ly (xAcme: ComponentType | xAcme: ConnectorType | PortType |
xAcme: Rol eType | xAcme: PropertyType | Component | Connector | Port | xAcme: Rol e
| xAcme: Property | Topol ogy | xAcme: Representati on)+>
<! ATTLIST     xAcme: Fami ly i denti fi er ID #REQUIRED>

<! ELEMENT     xAcme: Representati on (Archi tecture, Topol ogy?)>
<! ATTLIST     xAcme: Representati on i denti fi er ID #REQUIRED
              name CDATA #IMPLIED>

<! ELEMENT     xDarwi n: ComponentType (Component*)>
<! ATTLIST     xDarwi n: ComponentType i denti fi er ID #REQUIRED>

<! ELEMENT     xDarwi n: Inst  EMPTY>
<! ATTLIST     xDarwi n: Inst  name CDATA #REQUIRED
              prop CDATA #IMPLIED
              type CDATA #REQUIRED>

```

A-2 Modeling Specification

A-2.1 Planner Architecture in C2

```

<?xml versi on="1.0" ?>
<xADL>

```

```

<ComponentType xC2: OS="Uni x"
xC2: ImplementationModule="c2.planner.ClockComponent" xC2: Language="Java"
name="ClockComponent">
<Method name="Tick" xC2: direction="provide" xC2: MaptoOper="op1"/>
<Method name="setClockSpeed" xC2: direction="provide" xC2: MaptoOper="op2">
<Parameter name="rate" type="Integer" xC2: MaptoVar="op2.r" />
</Method>
<xC2: Behaviour>
<xC2: State> <xC2: Variable name="time" type="Integer" uid="time"/> </xC2: State>
<xC2: Invariant>
<xC2: Expression op1="speed" optype="eqgreater" op2="0" />
</xC2: Invariant>
<xC2: Operation name="op1" direction="provide" uid="op1">
<xC2: Post>
<xC2: Expression op1="time" optype="addition" op2="1" />
/xC2: Post>
</xC2: Operation>
<xC2: Operation name="op2" direction="provide" uid="op2">
<xC2: Let>
<xC2: Variable name="r" type="Integer" uid="op2.r" />
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="speed" optype="equalsto" op2="r" />
</xC2: Post>
</xC2: Operation>
</xC2: Behaviour>
</ComponentType>

```

```

<ComponentType xC2: OS="Uni x"
xC2: ImplementationModule="c2.planner.DeliverPortComponent" C2: Language="Java"
name="DeliverPortComponent">
<Method name="newShipment" xC2: direction="provide" xC2: MaptoOper="op_newshp">
<Parameter name="port" type="PortID" xC2: MaptoVar="op_newshp.pid" />
<Parameter name="shp" type="ShipmentType" xC2: MaptoVar="op_newshp.shp" />
</Method>
<Method name="unloadShipment" xC2: direction="provide" xC2: MaptoOper="op_unload">
<Parameter name="port" type="PortID" xC2: MaptoVar="op_unload.pid" />
<Parameter name="shp" type="Integer" xC2: MaptoVar="op_unload.shp" />
<Parameter type="ShipmentType" role="out" xC2: MaptoVar="op_unload.shp_by_sid" />
</Method>
<Method name="getDeliverPorts" xC2: direction="provide"
xC2: MaptoOper="op_getprt">
<Parameter type="DeliverPortTypeSet" role="out" xC2: MaptoVar="op_getprt.ports"
/>
</Method>
<Method name="Tick" xC2: direction="require" xC2: MaptoOper="or_timinc"/>

<xC2: Behaviour>
<xC2: State> <xC2: Variable name="ports" type="IntegerSet" uid="ports"/>
</xC2: State>
<xC2: Invariant>
<xC2: Expression op1="ports" optype="seteqgreater" op2="0" />
</xC2: Invariant>

<xC2: Operation name="op_newshp" direction="provide" uid="op_newshp">
<xC2: Let>
<xC2: Variable name="pid" type="Integer" uid="op_newshp.pid"/>
<xC2: Variable name="shp" type="ShipmentType" uid="op_newshp.shp"/>
</xC2: Let>
<xC2: Pre>
<xC2: Expression op1="pid" optype="setequalsto" op2="ports" />
<xC2: Expression op1="pid" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expression op1="ports" optype="setequalsto" op2="ports" />
</xC2: Post>

```

```

</xC2: Operati on>

<xC2: Operati on name="op_unl oad" di recti on="provi de" ui d="op_unl oad">
<xC2: Let>
<xC2: Vari abl e name="pi d" type="Integer" ui d="op_unl oad. pi d"/>
<xC2: Vari abl e name="si d" type="Integer" ui d="op_unl oad. si d"/>
<xC2: Vari abl e name="shp_by_si d" type="Shi pmentType" ui d="op_unl oad. shp_by_si d"/>
</xC2: Let>
<xC2: Pre>
<xC2: Expressi on op1="pi d" optype="seteql ess" op2="ports" />
<xC2: Expressi on op1="pi d" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expressi on op1="ports" optype="setequal sto" op2="ports" />
<xC2: Expressi on op1="resul t" optype="equal sto" op2="shp_si d" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="op_getprt" di recti on="provi de" ui d="op_getprt">
<xC2: Post>
<xC2: Expressi on op1="resul t" optype="equal sto" op2="ports" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="or_ti mi nc" di recti on="requi re" ui d="or_ti mi nc">
<xC2: Let>
<xC2: Vari abl e name="ti me" type="STATEVARI ABLE" ui d="or_ti mi nc. ti me"/>
</xC2: Let>
<xC2: Post>
<xC2: Expressi on op1="ti me" optype="addi ti on" op2="1" />
</xC2: Post>
</xC2: Operati on>

</xC2: Behavi our>
</ComponentType>

<ComponentType xC2: OS="Uni x"
xC2: Impl ementati onModul e="c2. pl anner. Vehi cl eComponent" xC2: Language="Java"
name="Vehi cl eComponent" >

<Method name="addShi pment" xC2: di recti on="provi de" xC2: MaptoOper="op_addshp">
<Parameter name="veh" type="Vehi cl eID" xC2: MaptoVar="op_addshp. vi d" />
<Parameter name="shp" type="Shi pmentType" xC2: MaptoVar="op_addshp. shp" />
</Method>
<Method name="unl oadShi pment" xC2: di recti on="provi de" xC2: MaptoOper="op_unl oad">
<Parameter name="veh" type="Vehi cl eID" xC2: MaptoVar="op_unl oad. vi d" />
</Method>
<Method name="getVehi cl es" xC2: di recti on="provi de" xC2: MaptoOper="op_getveh">
<Parameter type="Vehi cl eTypeSet" rol e="out" xC2: MaptoVar="op_getveh. vehi cl es" />
</Method>
<Method name="Ti ck" xC2: di recti on="requi re" xC2: MaptoOper="or_ti mi nc"/>

<xC2: Behavi our>
<xC2: State> <xC2: Vari abl e name="vehi cl es" type="Vehi cl eTypeSet" ui d="vehi cl es"/>
</xC2: State>
<xC2: Invari ant>
<xC2: Expressi on op1="vehi cl es" optype="seteqgreater" op2="0" />
</xC2: Invari ant>

<xC2: Operati on name="op_addshp" di recti on="provi de" ui d="op_addshp">
<xC2: Let>
<xC2: Vari abl e name="vi d" type="Integer" ui d="op_addshp. vi d"/>
<xC2: Vari abl e name="shp" type="Shi pmentType" ui d="op_addshp. shp"/>
</xC2: Let>
<xC2: Pre>
<xC2: Expressi on op1="vehi cl es" optype="seteqgreater" op2="vi d" />

```

```

<xC2: Expression op1="vi d" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expression op1="vehic les" optype="setequal sto" op2="vehic les" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="op_unl oad" di recti on="provi de" ui d="op_unl oad">
<xC2: Let>
<xC2: Variabl e name="vi d" type="Integer" ui d="op_unl oad. vi d"/>
</xC2: Let>
<xC2: Pre>
<xC2: Expression op1="vehic les" optype="seteqgreater" op2="vi d" />
<xC2: Expression op1="vi d" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expression op1="vehic les" optype="setequal sto" op2="vehic les" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="op_getveh" di recti on="provi de" ui d="op_getveh">
<xC2: Post>
<xC2: Expression op1="resul t" optype="equal sto" op2="vehic les" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="or_timi nc" di recti on="requi re" ui d="or_timi nc">
<xC2: Let>
<xC2: Variabl e name="ti me" type="STATEVARI ABLE" ui d="or_timi nc. ti me"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="ti me" optype="addi ti on" op2="1" />
</xC2: Post>
</xC2: Operati on>

</xC2: Behavi our>
</ComponentType>

<ComponentType xC2: OS="Uni x"
xC2: Implementati onModul e="c2. pl anner. WarehouseComponent" xC2: Language="Java"
name="WarehouseComponent" />

<Method name="addShi pment" xC2: di recti on="provi de" xC2: MaptoOper="op_addshp">
<Parameter name="wh" type="Warehou seID" xC2: MaptoVar="op_addshp. wi d"/>
<Parameter name="shp" type="Shi pmentType" xC2: MaptoVar="op_addshp. shp" />
</Method>
<Method name="unl oadShi pment" xC2: di recti on="provi de" xC2: MaptoOper="op_unl oad">
<Parameter name="wh" type="Vehi cl elD" xC2: MaptoVar="op_unl oad. wi d" />
</Method>
<Method name="getWarehouses" xC2: di recti on="provi de" xC2: MaptoOper="op_getwhs">
<Parameter type="Warehou seTypeSet" rol e="out"
xC2: MaptoVar="op_getwhs. warehouses" />
</Method>

<xC2: Behavi our>
<xC2: State> <xC2: Variabl e name="warehouses" type="Warehou seTypeSet"
ui d="warehouses"/> </xC2: State>
<xC2: Invari ant>
<xC2: Expression op1="warehouses" optype="seteqgreater" op2="0" />
</xC2: Invari ant>

<xC2: Operati on name="op_addshp" di recti on="provi de" ui d="op_addshp">
<xC2: Let>
<xC2: Variabl e name="wi d" type="Integer" ui d="op_addshp. wi d"/>
<xC2: Variabl e name="shp" type="Shi pmentType" ui d="op_addshp. shp"/>
</xC2: Let>

```

```

<xC2: Pre>
<xC2: Expressi on op1="warehouses" optype="seteqgreater" op2="wi d" />
<xC2: Expressi on op1="wi d" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expressi on op1="warehouses" optype="setequal sto" op2="warehouses" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="op_unl oad" di recti on="provi de" ui d="op_unl oad">
<xC2: Let>
<xC2: Vari abl e name="wi d" type="Integer" ui d="op_unl oad. wi d"/>
</xC2: Let>
<xC2: Pre>
<xC2: Expressi on op1="warehouses" optype="seteqgreater" op2="wi d" />
<xC2: Expressi on op1="wi d" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expressi on op1="warehouses" optype="setequal sto" op2="warehouses" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="op_getwhs" di recti on="provi de" ui d="op_getwhs">
<xC2: Post>
<xC2: Expressi on op1="resul t" optype="equal sto" op2="warehouses" />
</xC2: Post>
</xC2: Operati on>

</xC2: Behavi our>
</ComponentType>

<ComponentType xC2: OS="Uni x"
xC2: Impl ementati onModul e="c2. pl anner. NextShi pmentComponent" xC2: Language="Java"
name="NextShi pmentComponent" />

<Method name="enterNewShi pment" xC2: di recti on="provi de"
xC2: MaptoOper="op_setnsh">
<Parameter name="port" type="PortID" xC2: MaptoVar="op_setnsh. pi d" />
<Parameter name="shp" type="Shi pmentType" xC2: MaptoVar="op_setnsh. shp" />
</Method>
<Method name="wai tForShi pment" xC2: di recti on="provi de"
xC2: MaptoOper="op_wai tsh"/>
<Method name="newShi pment" xC2: di recti on="requi re" xC2: MaptoOper="or_newshp">
<Parameter name="port" type="PortID" xC2: MaptoVar="or_newshp. pi d" />
<Parameter name="shp" type="Shi pmentType" xC2: MaptoVar="or_newshp. shp" />
</Method>
<Method name="Ti ck" xC2: di recti on="requi re" xC2: MaptoOper="or_ti mi nc"/>

<xC2: Behavi our>
<xC2: State> <xC2: Vari abl e name="shi p_ti mer" type="Integer" ui d="shi p_ti mer"/>
</xC2: State>
<xC2: Invari ant>
<xC2: Expressi on op1="shi p_ti mer" optype="eqgreater" op2="0" />
</xC2: Invari ant>

<xC2: Operati on name="op_setnsh" di recti on="provi de" ui d="op_setnsh">
<xC2: Let>
<xC2: Vari abl e name="pi d" type="Integer" ui d="op_setnsh. pi d"/>
<xC2: Vari abl e name="shp" type="Shi pmentType" ui d="op_setnsh. shp"/>
</xC2: Let>
<xC2: Post>
<xC2: Expressi on op1="shi p_ti mer" optype="equal sto" op2="0" />
</xC2: Post>
</xC2: Operati on>

```

```

<xC2: Operation name="op_wai tsh" di recti on="provi de" ui d="op_wai tsh">
<xC2: Post>
<xC2: Expressi on op1="shi p_ti mer" optype="addi ti on" op2="1" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="or_newshp" di recti on="requi re" ui d="or_newshp">
<xC2: Let>
<xC2: Vari abl e name="pi d" type="Integer" ui d="or_newshp. pi d" />
<xC2: Vari abl e name="shp" type="Shi pmentType" ui d="or_newshp. shp" />
<xC2: Vari abl e name="ports" type="STATEVARI ABLE" ui d="or_newshp. ports" />
</xC2: Let>
<xC2: Pre>
<xC2: Expressi on op1="pi d" optype="seteql ess" op2="ports" />
<xC2: Expressi on op1="pi d" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expressi on op1="ports" optype="equal sto" op2="ports" />
</xC2: Post>
</xC2: Operati on>

<xC2: Operati on name="or_ti mi nc" di recti on="requi re" ui d="or_ti mi nc">
<xC2: Let>
<xC2: Vari abl e name="ti me" type="STATEVARI ABLE" ui d="or_ti mi nc. ti me" />
</xC2: Let>
<xC2: Post>
<xC2: Expressi on op1="ti me" optype="addi ti on" op2="1" />
</xC2: Post>
</xC2: Operati on>

</xC2: Behavi our>
</ComponentType>

<ComponentType xC2: OS="Uni x"
xC2: Impl ementati onModul e="c2. pl anner. Del i veryPortArti st" xC2: Language="Java"
name="Del i veryPortArti st" />

<Method name="createVi ewport" xC2: di recti on="provi de" xC2: MaptoOper="op_newpvt">
<Parameter type="Vi ewportType" rol e="out" xC2: MaptoVar="op_newpvt. port" />
</Method>
<Method name="newPortLi st" xC2: di recti on="provi de" xC2: MaptoOper="op_prtl st">
<Parameter type="Stri ng" rol e="out" xC2: MaptoVar="op_prtl st. port_i nfo" />
</Method>
<Method name="sel ectShi pment" xC2: di recti on="provi de" xC2: MaptoOper="op_sel shp">
<Parameter name="port" type="PortID" xC2: MaptoVar="op_sel shp. pi d" />
<Parameter name="shp" type="Shi pmentID" xC2: MaptoVar="op_sel shp. si d" />
</Method>
<Method name="desel ectShi pment" xC2: di recti on="provi de"
xC2: MaptoOper="op_desshp">
<Parameter name="port" type="PortID" xC2: MaptoVar="op_desshp. pi d" />
<Parameter name="shp" type="Shi pmentID" xC2: MaptoVar="op_desshp. si d" />
</Method>
<Method name="unl oadShi pment" xC2: di recti on="requi re" xC2: MaptoOper="or_unl oad">
<Parameter name="port" type="Integer" xC2: MaptoVar="or_unl oad. pi d" />
<Parameter type="Shi pmentType" rol e="out" xC2: MaptoVar="or_unl oad. pi d" />
</Method>
<Method name="getDel i veryPorts" xC2: di recti on="requi re"
xC2: MaptoOper="or_getprt">
<Parameter type="Del i veryPortTypeSet" rol e="out" xC2: MaptoVar="or_getprt. ports"
/>
</Method>

<xC2: Behavi our>

```

```

<xC2: State>
<xC2: Variable name="vport" type="ViewportType" uid="vport">
<xC2: Variable name="selections" type="IntegerSet" uid="selections">
</xC2: Variable>
</xC2: State>
<xC2: Invariant>
<xC2: Expression op1="selections" optype="seteqgreater" op2="0" />
</xC2: Invariant>

<xC2: Operation name="op_newpvt" direction="provide" uid="op_newpvt">
<xC2: Post>
<xC2: Expression op1="result" optype="equalsto" op2="vport" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="op_prtlst" direction="provide" uid="op_prtlst">
<xC2: Let>
<xC2: Variable name="port_info" type="String" uid="op_prtlst.port_info"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="result" optype="equalsto" op2="port_info" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="op_selshp" direction="provide" uid="op_selshp">
<xC2: Let>
<xC2: Variable name="pid" type="Integer" uid="op_selshp.pid"/>
<xC2: Variable name="sid" type="Integer" uid="op_selshp.sid"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="selections" optype="setaddition" op2="1" />
<xC2: Expression op1="10*pid+sid" optype="notation" op2="selections" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="op_desshp" direction="provide" uid="op_desshp">
<xC2: Let>
<xC2: Variable name="pid" type="Integer" uid="op_desshp.pid"/>
<xC2: Variable name="sid" type="Integer" uid="op_desshp.sid"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="selections" optype="setsubtraction" op2="1" />
<xC2: Expression op1="10*pid+sid" optype="notation" op2="selections" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="or_unload" direction="require" uid="or_unload">
<xC2: Let>
<xC2: Variable name="pid" type="Integer" uid="or_unload.pid"/>
<xC2: Variable name="ports" type="STATEVARIABLE" uid="or_unload.ports"/>
<xC2: Variable name="shp_by_sid" type="ShipmentType" uid="or_unload.shp_by_sid"/>
</xC2: Let>
<xC2: Pre>
<xC2: Expression op1="ports" optype="seteqgreater" op2="pid" />
<xC2: Expression op1="pid" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expression op1="ports" optype="setequalsto" op2="ports" />
<xC2: Expression op1="result" optype="equalsto" op2="shp_by_sid" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="or_getprt" direction="require" uid="or_getprt">
<xC2: Let>
<xC2: Variable name="ports" type="STATEVARIABLE" uid="or_getprt.ports"/>
</xC2: Let>

```

```

<xC2: Post>
<xC2: Expression op1="result" optype="equalsto" op2="ports" />
</xC2: Post>
</xC2: Operation>

</xC2: Behaviour>
</ComponentType>

<ComponentType xC2: OS="Unix"
xC2: ImplementationModule="c2.planner.VehicleArtisticComponent" xC2: Language="Java"
name="VehicleArtisticComponent" />

<Method name="createViewport" xC2: direction="provide" xC2: MaptoOper="op_newpvt">
<Parameter type="ViewportType" role="out" xC2: MaptoVar="op_newpvt.vport" />
</Method>
<Method name="newVehicleList" xC2: direction="provide" xC2: MaptoOper="op_vhlist">
<Parameter type="String" role="out" xC2: MaptoVar="op_vhlist.veh_info" />
</Method>
<Method name="selectVehicle" xC2: direction="provide" xC2: MaptoOper="op_selveh">
<Parameter name="veh" type="VehicleID" xC2: MaptoVar="op_selveh.vid" />
</Method>
<Method name="deselectVehicle" xC2: direction="provide"
xC2: MaptoOper="op_desveh">
<Parameter name="veh" type="VehicleID" xC2: MaptoVar="op_desveh.vid" />
</Method>
<Method name="unloadShipment" xC2: direction="require" xC2: MaptoOper="or_unload">
<Parameter name="veh" type="VehicleID" xC2: MaptoVar="or_unload.vid" />
</Method>
<Method name="getVehicles" xC2: direction="require" xC2: MaptoOper="or_getveh">
<Parameter type="VehicleTypeSet" role="out" xC2: MaptoVar="or_getveh.vehicles" />
</Method>

<xC2: Behaviour>
<xC2: State>
<xC2: Variable name="vport" type="ViewportType" uid="vport">
<xC2: Variable name="selection" type="Integer" uid="selection">
</xC2: Variable>
</xC2: State>
<xC2: Invariant>
<xC2: Expression op1="selection" optype="eqgreater" op2="0" />
</xC2: Invariant>

<xC2: Operation name="op_newpvt" direction="provide" uid="op_newpvt">
<xC2: Post>
<xC2: Expression op1="result" optype="equalsto" op2="vport" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="op_vhlist" direction="provide" uid="op_vhlist">
<xC2: Let>
<xC2: Variable name="veh_info" type="String" uid="op_vhlist.veh_info"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="result" optype="equalsto" op2="veh_info" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="op_selveh" direction="provide" uid="op_selveh">
<xC2: Let>
<xC2: Variable name="vid" type="Integer" uid="op_selveh.pid"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="vid" optype="equalsto" op2="selection" />
</xC2: Post>
</xC2: Operation>

```

```

<xC2: Operation name="op_desveh" direction="provide" uid="op_desveh">
<xC2: Let>
<xC2: Variable name="vid" type="Integer" uid="op_desveh.vid"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="vid" optype="notequalsto" op2="selection" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="or_unload" direction="require" uid="or_unload">
<xC2: Let>
<xC2: Variable name="vid" type="Integer" uid="or_unload.vid"/>
<xC2: Variable name="vehicles" type="STATEVARIABLE" uid="or_unload.vehicles"/>
</xC2: Let>
<xC2: Pre>
<xC2: Expression op1="vehicles" optype="seteqgreater" op2="vid" />
<xC2: Expression op1="vid" optype="greater" op2="0" />
</xC2: Pre>
<xC2: Post>
<xC2: Expression op1="vehicles" optype="setequalsto" op2="vehicles" />
</xC2: Post>
</xC2: Operation>

<xC2: Operation name="or_getveh" direction="require" uid="or_getveh">
<xC2: Let>
<xC2: Variable name="vehicles" type="STATEVARIABLE" uid="or_getveh.vehicles"/>
</xC2: Let>
<xC2: Post>
<xC2: Expression op1="result" optype="equalsto" op2="vehicles" />
</xC2: Post>
</xC2: Operation>

</xC2: Behaviour>
</ComponentType>

<ComponentType xC2: OS="Unix"
xC2: ImplementationModule="c2.planner.WarehouseArti stComponent"
xC2: Language="Java" name="WarehouseArti stComponent"/>

<Method name="createVie wport" xC2: direction="provide" xC2: MaptoOper="op_newpvt">
<Parameter type="Vie wportType" role="out" xC2: MaptoVar="op_newpvt.vport" />
</Method>
<Method name="newWarehouseLi st" xC2: direction="provide"
xC2: MaptoOper="op_whli st">
<Parameter type="String" role="out" xC2: MaptoVar="op_whli st.wh_i nfo" />
</Method>
<Method name="selectWarehouse" xC2: direction="provide"
xC2: MaptoOper="op_sel whs">
<Parameter name="wh" type="WarehouseI D" xC2: MaptoVar="op_sel whs.wi d" />
</Method>
<Method name="desel ectWarehouse" xC2: direction="provide"
xC2: MaptoOper="op_deswhs">
<Parameter name="wh" type="WarehouseI D" xC2: MaptoVar="op_deswhs.wi d" />
</Method>
<Method name="unl oadShi pment" xC2: direction="require" xC2: MaptoOper="or_unl oad">
<Parameter name="wh" type="WarehouseI D" xC2: MaptoVar="or_unl oad.wi d" />
</Method>
<Method name="getWarehouses" xC2: direction="require" xC2: MaptoOper="or_getwhs">
<Parameter type="WarehouseTypeSet" role="out"
xC2: MaptoVar="or_getwhs.warehouses" />
</Method>

<xC2: Behaviour>
<xC2: State>
<xC2: Variable name="vport" type="Vie wportType" uid="vport">

```

```
<xC2:Variable name="selection" type="Integer" uid="selection">
</xC2:Variable>
</xC2:State>
<xC2:Invariant>
<xC2:Expression op1="selection" optype="eqgreater" op2="0" />
</xC2:Invariant>

<xC2:Operation name="op_newpvt" direction="provide" uid="op_newpvt">
<xC2:Post>
<xC2:Expression op1="result" optype="equalsto" op2="vport" />
</xC2:Post>
</xC2:Operation>

<xC2:Operation name="op_whlist" direction="provide" uid="op_whlist">
<xC2:Let>
<xC2:Variable name="wh_info" type="String" uid="op_whlist.wh_info"/>
</xC2:Let>
<xC2:Post>
<xC2:Expression op1="result" optype="equalsto" op2="wh_info" />
</xC2:Post>
</xC2:Operation>

<xC2:Operation name="op_selwhs" direction="provide" uid="op_selwhs">
<xC2:Let>
<xC2:Variable name="wid" type="Integer" uid="op_selwhs.wid"/>
</xC2:Let>
<xC2:Post>
<xC2:Expression op1="wid" optype="equalsto" op2="selection" />
</xC2:Post>
</xC2:Operation>

<xC2:Operation name="op_deswhs" direction="provide" uid="op_deswhs">
<xC2:Let>
<xC2:Variable name="wid" type="Integer" uid="op_deswhs.wid"/>
</xC2:Let>
<xC2:Post>
<xC2:Expression op1="wid" optype="notequalsto" op2="selection" />
</xC2:Post>
</xC2:Operation>

<xC2:Operation name="or_unload" direction="require" uid="or_unload">
<xC2:Let>
<xC2:Variable name="wid" type="Integer" uid="or_unload.wid"/>
<xC2:Variable name="warehouses" type="STATEVARIABLE"
uid="or_unload.warehouses"/>
</xC2:Let>
<xC2:Pre>
<xC2:Expression op1="warehouses" optype="seteqgreater" op2="wid" />
<xC2:Expression op1="wid" optype="greater" op2="0" />
</xC2:Pre>
<xC2:Post>
<xC2:Expression op1="warehouses" optype="setequalsto" op2="warehouses" />
</xC2:Post>
</xC2:Operation>

<xC2:Operation name="or_getwhs" direction="require" uid="or_getwhs">
<xC2:Let>
<xC2:Variable name="warehouses" type="STATEVARIABLE"
uid="or_getwhs.warehouses"/>
</xC2:Let>
<xC2:Post>
<xC2:Expression op1="result" optype="equalsto" op2="warehouses" />
</xC2:Post>
</xC2:Operation>

</xC2:Behaviour>
```

```

</ComponentType>

<ComponentType xC2: OS="Uni x"
xC2: ImplementationModule="c2.comp.graphics.GraphicsBinding" xC2: Language="Java"
name="GraphicsBinding" />

<ConnectorType xC2: OS="Uni x" xC2: Language="Java" name="Bus1" />
<ConnectorType xC2: OS="Uni x" xC2: Language="Java" name="Bus2" />
<ConnectorType xC2: OS="Uni x" xC2: Language="Java" name="Bus3" />
<ConnectorType xC2: OS="Uni x" xC2: Language="Java" name="Bus4" />
<ConnectorType xC2: OS="Uni x" xC2: Language="Java" name="Bus5" />

<Architecture name="Planner" >
<Component name="Clock1"> <Supports type="ClockComponent"/> </Component>
<Component name="DelPort2"> <Supports type="DeliverPortComponent"/>
</Component>
<Component name="Vehicle3" > <Supports type="VehicleComponent"/> </Component>
<Component name="Warehouse4"> <Supports type="WarehouseComponent"/> </Component>
<Component name="NextShipment5"> <Supports type="NextShipmentComponent"/>
</Component>
<Component name="DelPortArtist6"> <Supports type="DeliverPortArtistComponent"/>
</Component>
<Component name="VehicleArtist7"> <Supports type="VehicleArtistComponent"/>
</Component>
<Component name="WarehouseArtist8"> <Supports type="WarehouseArtistComponent"/>
</Component>
<Component name="CargoRouter9"> <Supports type="CargoRouterComponent"/>
</Component>
<Component name="GraphicsBinding10"> <Supports type="GraphicsBinding"/>
</Component>

<Connector name="Bus1"> <Supports type="Bus1" /> </Connector>
<Connector name="Bus2"> <Supports type="Bus2" /> </Connector>
<Connector name="Bus3"> <Supports type="Bus3" /> </Connector>
<Connector name="Bus4"> <Supports type="Bus4" /> </Connector>
<Connector name="Bus5"> <Supports type="Bus5" /> </Connector>

<Topology name="Planner" >
<Link to="Bus1" from="Clock1" />
<Link to="Bus2" from="Bus1" />
<Link to="DelPort2" from="Bus1" />
<Link to="Vehicle3" from="Bus1" />
<Link to="Bus2" from="DelPort2" />
<Link to="Bus2" from="Vehicle3" />
<Link to="Bus2" from="Warehouse4" />
<Link to="NextShipment5" from="Bus2" />
<Link to="Bus3" from="Bus2" />
<Link to="DelPortArtist6" from="Bus3" />
<Link to="VehicleArtist7" from="Bus3" />
<Link to="WarehouseArtist8" from="Bus3" />
<Link to="Bus4" from="DelPortArtist6" />
<Link to="Bus4" from="VehicleArtist7" />
<Link to="Bus4" from="WarehouseArtist8" />
<Link to="CargoRouter9" from="Bus4" />
<Link to="Bus5" from="CargoRouter9" />
<Link to="GraphicsBinding10" from="Bus5" />
</Topology>

</Architecture>
</xADL>

```

A-2.2 Compiler Architecture in SADL

```
<?xml version="1.0" ?>
```

```

<xADL>
<Architecture name="Compiler" >

<Port type="in" name="char_import" datatype="SEQ(character)" />
<Port type="out" name="code_oport" datatype="code" />

<xSADL: Importing from="compiler_types">
<xSADL: ImportList> character </xSADL: ImportList>
<xSADL: ImportList> code </xSADL: ImportList>
<xSADL: ImportList> token </xSADL: ImportList>
<xSADL: ImportList> binding </xSADL: ImportList>
<xSADL: ImportList> ast </xSADL: ImportList>
</xSADL: Importing>

<xSADL: Importing from="Functional_style">
<xSADL: ImportList> Function </xSADL: ImportList>
</xSADL: Importing>

<xSADL: Importing from="Process_Pipeline_style">
<xSADL: ImportList> Pipe </xSADL: ImportList>
<xSADL: ImportList> Finite_Stream </xSADL: ImportList>
<xSADL: ImportList> Connects </xSADL: ImportList>
</xSADL: Importing>

<xSADL: Importing from="Shared_Memory_style">
<xSADL: ImportList> Variable </xSADL: ImportList>
<xSADL: ImportList> Reads </xSADL: ImportList>
<xSADL: ImportList> Writes </xSADL: ImportList>
</xSADL: Importing>

<xSADL: Importing from="Batch_Sequential_style">
<xSADL: ImportList> Start_After_Finish_Of </xSADL: ImportList>
</xSADL: Importing>

<Component name="Lexical_analyzer_module" type="architecture" >
<Port type="in" name="char_import" datatype="SEQ(character)" />
<Port type="out" name="token_oport" datatype="Finite_Stream(token)" />
</Component>
<Component name="parser" type="function" >
<Port type="in" name="token_import" datatype="Finite_Stream(token)" />
</Component>
<Component name="analyzer_optimizer" type="function" />
<Component name="code_generator" type="function" >
<Port type="out" name="code_oport" datatype="code" />
</Component>
<Component name="abstract_syntax_tree" type="variable" datatype="ast"/>

<Connector name="token_pipe" type="Pipe" datatype="Finite_Stream(token)" />

<Topology name="Compiler" >

<Link name="token_flow" conname="token_pipe" >
<Port type="in" name="token_import" />
<Port type="out" name="token_oport"/>
</Link>

<xSADL: Constraint name="read_bind" type="reads"
source="analyzer_optimizer" destination="Lexical_analyzer_module!symbol_table"
/>
<xSADL: Constraint name="write_base_ast" type="writes"
source="parser" destination="abstract_syntax_tree" />
<xSADL: Constraint name="read_base_ast" type="reads"
source="analyzer_optimizer" destination="abstract_syntax_tree" />
<xSADL: Constraint name="write_full_ast" type="writes"
source="analyzer_optimizer" destination="abstract_syntax_tree" />

```

```
<xSADL:Constraint name="read_full_ast" type="reads"
source="code_generator" destination="abstract_syntax_tree" />
<xSADL:Constraint name="precedence_1" type="Starts_After_Finish_of"
source="analyzer_optimizer" destination="parser" />
<xSADL:Constraint name="precedence_2" type="Starts_After_Finish_of"
source="code_generator" destination="analyzer_optimizer" />
</Topology>

</Architecture>
</xADL>
```

A-2.3 Client Server Architecture in Acme

```
<?xml version="1.0" ?>
<xADL>
<Architecture name="simple-cs" >

<Component name="client">
<Port name="send-request">
</Port>
</Component>

<Component name="server">
<Port name="receive-request">
</Port>
</Component>

<Connector name="rpc">
<xAcmeRole name="caller">
</xAcmeRole>
<xAcmeRole name="callee">
</xAcmeRole>
</Connector>

<Topology name="simple-cs">
<Link from="client.send-request" to="rpc.caller">
</Link>
<Link from="server.receive-request" to="rpc.callee">
</Link>

</Topology>
</Architecture>
</xADL>
```

A-2.4 Client Server Architecture with Properties in Acme

```
<?xml version="1.0" ?>
<xADL>
<Architecture name="simple-cs2" >

<Component name="client">
<Port name="send-request">
</Port>
<Properties name="Aesop-style" type="style-id" value="client-server">
</Properties>
<Properties name="Unicon-style" type="style-id" value="cs">
</Properties>
<Properties name="source-code" type="external" value="CODE-LIB/client.c">
</Properties>
```

```
</Component>

<Component name = "server">
<Port name = "receive-request">
</Port>
<Properties name = "idempotence" type = "boolean" value = "true">
</Properties>
<Properties name = "max-concurrent-clients" type = "integer" value="1">
</Properties>
<Properties name = "source-code" type = "external" value = "CODE-LIB/server.c">
</Properties>
</Component>

<Connector name ="rpc">
<xAcmeRole name="caller">
</xAcmeRole>
<xAcmeRole name="callee">
</xAcmeRole>
<Properties name = "synchronous" type = "boolean" value = "true">
</Properties>
<Properties name = "max-roles" type = "integer" value = "2">
</Properties>
<Properties name = "protocol" type = "Wright" value = "... ">
</Properties>
</Connector>

<Topology name ="simple-cs">
<Link from="client.send-request" to="rpc.caller">
</Link>
<Link from="server.receive-request" to="rpc.callee">
</Link>

</Topology>
</Architecture>
</xADL>
```

A-2.5 Lazy Pipe Architecture in DARWIN

```
<?xml version="1.0" ?>
<xADL>
<Component name="lazypipe">
<Port type = "out" name ="input"></Port>
<Port type = "in" name ="output"></Port>
<xDarwinInst name ="H" type="Filter"></xDarwinInst>
<xDarwinInst name ="T" prop ="dyn" type="..."></xDarwinInst>
<Link from="input" to="H.prev"></Link>
<Link from="H.next" to="T.input"></Link>
<Link from="H.output" to="output"></Link>
<Link from="T.output" to="output"></Link>
<Port type = "in" name ="output"></Port>
<Port type = "out" name ="input"></Port>
</Component>
</xADL>
```

A-2.6 Client Server Architecture in Darwin

```
<?xml version="1.0" ?>
<xADL>
<Component name = "Client">
<Port type = "in" name = "r"></Port>
</Component>
```

```
<Component name="Server">
<Port type = "out" name = "p"></Port>
</Component>

<Component name = "System">
<xDarwi nI nst name = "A" type= "Cl ient"></xDarwi nI nst>
<xDarwi nI nst name = "B" type= "Server"></xDarwi nI nst>
<Link from = "A" to = "B"></Li nk>
<Port type = "in" name = "r"></Port>
<Port type = "out" name = "p"></Port>
</Component>
</xADL>
```

A-3 C2 Constraint Checker (Perl)

```
#assumes case sensitivity in .xml file for the time being, hence "me" different
from "Me"
while ($my = <> ) #read from redirected input .xml file
{
if ($my=~/\^s*<Component\s+. *name="(\w+)"/i) #get the component name, put in
an assoc. array
{
#pri nt "componame: $1\n";
$el ements{$1}=' component' ;
}
el si f ($my=~/\^s*<Connector\s+. *name="(\w+)"/i) #get the connector name, put in
an assoc. array
{
#pri nt "connecname: $1\n";
$el ements{$1}=' connector' ;
}
el si f ($my=~/\^s*<Li nk. *To="(\w+)". *From="(\w+)"/i) #get from and to names
{
# pri nt "from: $2 to: $1 \n";
push (@$l i nks{$2}, $1);
push (@$r l i nks{$1}, $2);
}
}
#####
#Check for three C2 constraints
#1) No component-component connection
#2) No multiple links attached to a component bottom
#3) No multiple links attached to a component top

foreach $link (sort keys(%links))
{
```

```
$suzunluk=@{$links{$link}};
for ($i=0; $i<$suzunluk; $i++)
{
    $oops=@{$links{$link}}[$i];
    #print "esas: $link    oops: $oops\n";
    #print "esast: $elements{$link}    oopst: $elements{$oops}\n";
    if (($elements{$link} eq 'component') && ($elements{$oops} eq 'component'))
    {
        print "C2Error: Trying to connect two components, $link and $oops\n";
    }
}
if (($elements{$link} eq 'component') && ($suzunluk>1))
{
    print "C2Error: Component $link detected to have multiple connections on its
bottom\n";
}
}
foreach $rlink (sort keys(%rlinks))
{
    $suzunluk=@{$rlinks{$rlink}};

    if (($elements{$rlink} eq 'component' ) && ($suzunluk>1))
    {
        print "C2Error: Component $rlink detected to have multiple connections on its
top\n";
    }
}
```