A Project in the ITL Pervasive Computing Portfolio

Applying ADLs to Assess
Emerging Industry Specifications for
Dynamic Discovery of Ad Hoc Network Services

Christopher Dabrowski and Kevin Mills

DARPA PI Meeting
January 31, 2001
Project Goals

1) Use ADLs and associated tools to **analyze Discovery Protocol specifications** to assess consistency and completeness wrt dynamic change conditions—**provide basis for gauges**.

2) **Compare and contrast emerging** commercial service **discovery technologies** with regard to function, structure, behavior, performance and scalability in the face of dynamic change.
Presentation Topics

- Planned Approach to Modeling and Analysis and Current Status
- Technical Discussion of Initial Progress
  - Generic and Specific UML Models Encompassing Jini, UPnP, & SLP
    - Rapide Model for Jini (90% complete)
- Upcoming Milestones and Planned Publications
Modeling Function, Structure, and Behavior

Objectives

1. Provide increased understanding of the competing dynamic discovery services emerging in industry
2. Develop metrics/gauges for comparative analysis of different approaches to dynamic discovery and for analyzing consistency and completeness of discovery protocols
3. Assess suitability of architecture description languages to model and analyze emerging dynamic discovery protocols

Technical Approach

- Develop ADL models from selected specifications for service discovery protocols and develop a suite of scenarios and topologies with which to exercise the ADL models
- Propose a set of invariant properties that all dynamic discovery protocols should satisfy
- Propose a set of metrics, based on partially ordered sets, with which to compare and contrast discovery protocols
- Analyze the ADL models for inconsistencies, to assess invariant satisfaction, and to compare and contrast protocols

Status as of January 31, 2001

- Developed a generic UML model encompassing the structure and function of Jini, UPnP, SLP, Bluetooth, and HAVi
- Projected specific UML models for Jini, UPnP, and SLP
- Developed a Rapide Model of Jini structure, function, and behavior (90% complete)
- Drafted a scenario language to drive the Rapide Jini Model; currently being implemented.
- Developed some initial invariants and constraints for Jini behavioral model
- Discovered a number of ambiguities and inconsistencies in Jini Specification V1.1
- Discovered a major architectural issue in the interaction between Jini directed discovery and multicast discovery

Products

- Rapide specifications of Jini, Universal Plug and Play (UPnP), and Service Location Protocol (SLP)
- Scenarios and topologies for evaluating discovery protocols
- Suggested invariant properties for service discovery protocols
- Suggested metrics, based on partially ordered sets (POSEts), for comparing and contrasting discovery protocols
- Paper identifying inconsistencies and ambiguities in Jini and UPnP and describing how they were found
- Paper proposing invariants for service discovery protocols, and evaluating how Jini, UPnP, and SLP fare
- Paper comparing and contrasting Jini, UPnP, and SLP at the level of POSET metrics
Benefits from Using Architecture, ADLs, & Tools

• **Represent essential complexity** with effective abstractions

• **Provide a framework and context**
  – to more easily **pinpoint** where **inconsistencies and ambiguities** may exist within software implementing specifications & to understand how they arise
  – to **compare and contrast** different discovery **protocols** (Jini, UP&P, SLP)
  – to **define gauges** that yield qualitative and quantitative measures
Generic UML Structural Model of Service Discovery Protocols
Architecture Description Languages and Tools

Allow us to model the essential complexity of discovery protocols, while ignoring the incidental complexity.

Jini documented in a 385 page specification; however, the document is static and thus captures only the normative complexity because most of the essential complexity arises through interactions among distributed, independently acting, Jini components.

Incidental complexity represented by the code: for example consider Core Jini – an 832 page commentary on the massive amount of Java code that comprises Jini, which also depends on complex underlying code for Remote Method Invocation, Distributed Events, Object Serialization, TCP/IP, UDP, HTTP, and Multicast Protocol Implementation.
Rapide, an Architecture Description Language and Tools
Developed for DARPA by Stanford

MODELING
ESSENTIAL COMPLEXITY

Specification of Rapide Architecture

Execute with Raptor Engine

Analyze Generated POSETs

Assess Invariant Satisfaction & Constraint Violations
Layered View of Prototype JINI Architecture in Rapide
Derived from SEI Architectural Layers Approach

Legend
Type of
Part of
Execute Architecture with the Raptor Engine
Drive Model Topology with Scenarios

> StartTime {NodeFail || NodeRecover} NodeID DelayTime.
> StartTime {LinkFail || LinkRestore} NodeID DelayTime FromNode ToNode.
> StartTime {MProbeFail || MProbeRestore} NodeID DelayTime FromNode ToNode.
> StartTime {GroupJoin || GroupLeave} NodeID DelayTime.
> StartTime {AddSCM || DeleteSCM} NodeID DelayTime.
> StartTime {AddService ChangeService} NodeID DelayTime ServiceTemplate ServiceAPI ServiceGUI LeaseTime DurationTime.
> StartTime DeleteService NodeID DelayTime ServiceID.
> StartTime FindService NodeID DelayTime SMNodeID.
> StartTime AddNotificationRequest NodeID DelayTime NotificationID ServiceTemplate Transitions LeaseTime DurationTime SCMID.
> StartTime DeleteNotificationRequest NodeID DelayTime NotificationID SCMID.
Analyze Invariant Satisfaction & Constraint Violations in Real-Time

**Sample Invariants**

\[
\begin{align*}
\circ (SM \leftrightsquigarrow SD \leftrightsquigarrow SCM): (SM,SD) & \triangleright SCM \text{ registered-services} \\
& \triangleright SCM \text{ SM discovered-SCMs}
\end{align*}
\]

\[
\begin{align*}
\circ (SU \leftrightsquigarrow NR \leftrightsquigarrow SCM): (SU,NR) & \triangleright SCM \text{ registered-notifications} \\
& \triangleright SCM \text{ SU discovered-SCMs}
\end{align*}
\]

- SM is Service Manager
- SD is Service Description
- SCM is Service Cache Manager
- SU is Service User
- NR is Notification Request
- Registered-services is a set of (SM,SD) pairs
- Registered-notifications is a set of (SU,NR) pairs
- Discovered-SCMs is a set of SCM

→ Invariants provide basis for defining *gauges* that provide qualitative measures of properties of a system
Analyze POSETs\rightarrow Off-Line to Compare and Contrast Behaviors Given a Congruent Topology and Scenario

Metrics Based on Numbers of Messages
- Message volume?
- Message intensity?

Metrics Based on Complexity
- Degree of dependency among messages?
- Rate of constraint and invariant violations?
- Rate of exceptions?

Metrics Based on Time
- Service latency?
- Service throughput?
- Recovery latency?

Metrics Based on Change
- Derivative of the message intensity?
- Derivative of the service throughput?
- Derivative of the service latency?

\rightarrow POSET analyses provide basis for defining gauges that provide quantitative measures of properties of a system
Schedule and Milestones

• FY 2001
  – Operational prototype of Jini & UPnP architectures
  – Report showing initial results of analysis of Jini and compare/contrast of Jini & UPnP; recommendations on ADLs.

• FY 2002
  – Formalization of quantitative & qualitative metrics to serve as basis for gauges; formalization of compare/contrast analysis
  – Expansion of operational prototype to incorporate metrics & resulting analysis as well as SLP (other protocols?)
  – Second report on results.
Generic UML Functional Model of Service Discovery Protocols

NOTE: This <<Interface>> exists only if there are no Cache Managers. The condition applies to SLP.

NOTE: If Cache MGRs are supported, Service Discovery may be "Cache MGR Discovery". If not, it may be "Service Listening".
UML Structural Model of Jini

SERVICE MANAGER
- discover Network Context()
- Cache Manager Discovery()
- Announce Service Processing()
- start Renewal Task()
- Service Manager()

<<repository entry>>
Service Repository
Contains
1
Aggregates
0..*

SERVICE PROVIDER
Identify
Type
API
GUI
Attributes
1

SERVICE USER
- discover Network Context()
- Service Discovery()
- start Renewal Task()
- Service User()

SERVICE CACHE MANAGER
- discover Network Context()
- activate Manager Discovery()
- activate Announce Processing()
- start Matching Task()
- start Aging Task()
- Service Cache Manager()

<<repository entry>>
Service Cache
Contains
0..1
Aggregates
0..*

<<repository entry>>
Notification Cache
Contains
0..1
Aggregates
0..*

<<repository entry>>
Notification Request
(from Data View)

0..*
0..*
0..*
0..*
0..*
0..*
0..*
UML Functional Model of Jini

NOTE: This <<Interface>> exists only if there are no Cache Managers. The condition applies to SLP.

NOTE: If Cache MGRs are supported, Service Discovery may be "Cache MGR Discovery". If not, it
Plan to Assess Scalability

• Use Rapide Models as a Basis to Construct Simulation Models for Jini, UPnP and SLP, Possibly using JavaSim (from Ohio State University) or SSFnet (from Rutgers)

• Use Results from Measurement Portion of the Project to Parameterize the Simulation Models of the Discovery Protocols

• Design Experiments to Assess the Effect of Large Service and Device Populations on Network Traffic