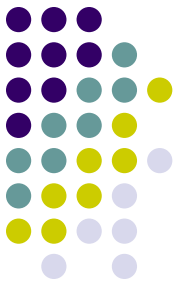


# **SERVICE CONTEXT MANAGEMENT IN SORCER**

**K.M.DIVYA DARSHAN**  
**TTU SORCER LAB**

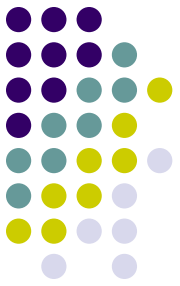
# Agenda



- 1. Terminology**
- 2. Background**
- 3. Problem statement**
- 4. Objective**
- 5. Methodology**
- 6. Schedule**
- 7. Benefits**

## Terminology :

- 1) **Exertion** – an act of requesting a service.
- 2) **Service Task** –
  - a) is a batch of instructions processing the same Service Context .
  - b) an elementary item of work.
- 3) **Service Job** –
  - a) is the analog of a procedure in conventional programming languages .
  - b) an composite item of work.
- 4) **Service Context** –
  - a) data that tasks and jobs work on .
  - b) an input and output parameter of any provider method .

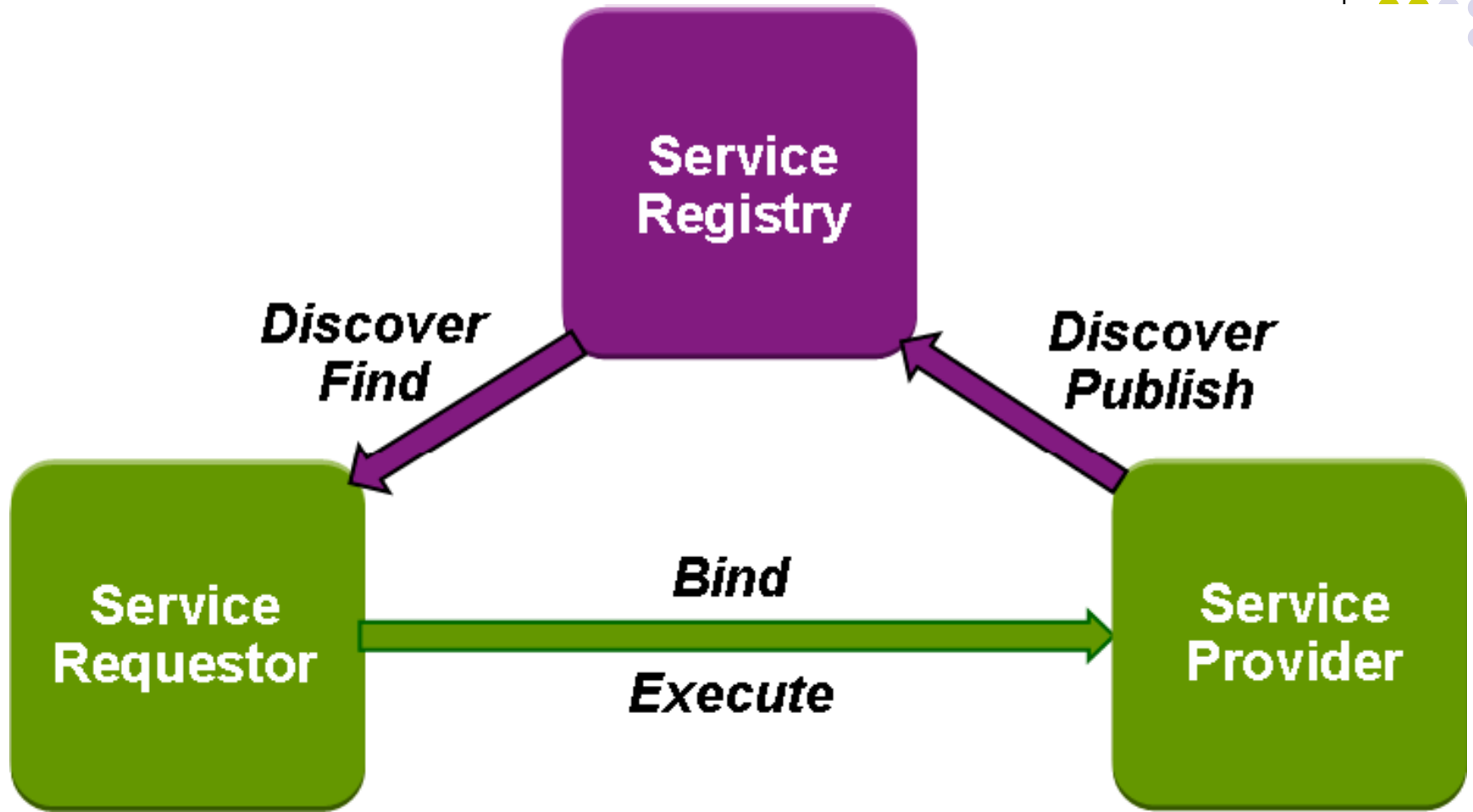
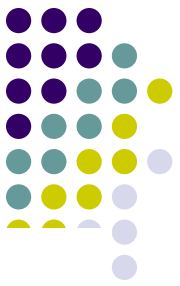


## Background

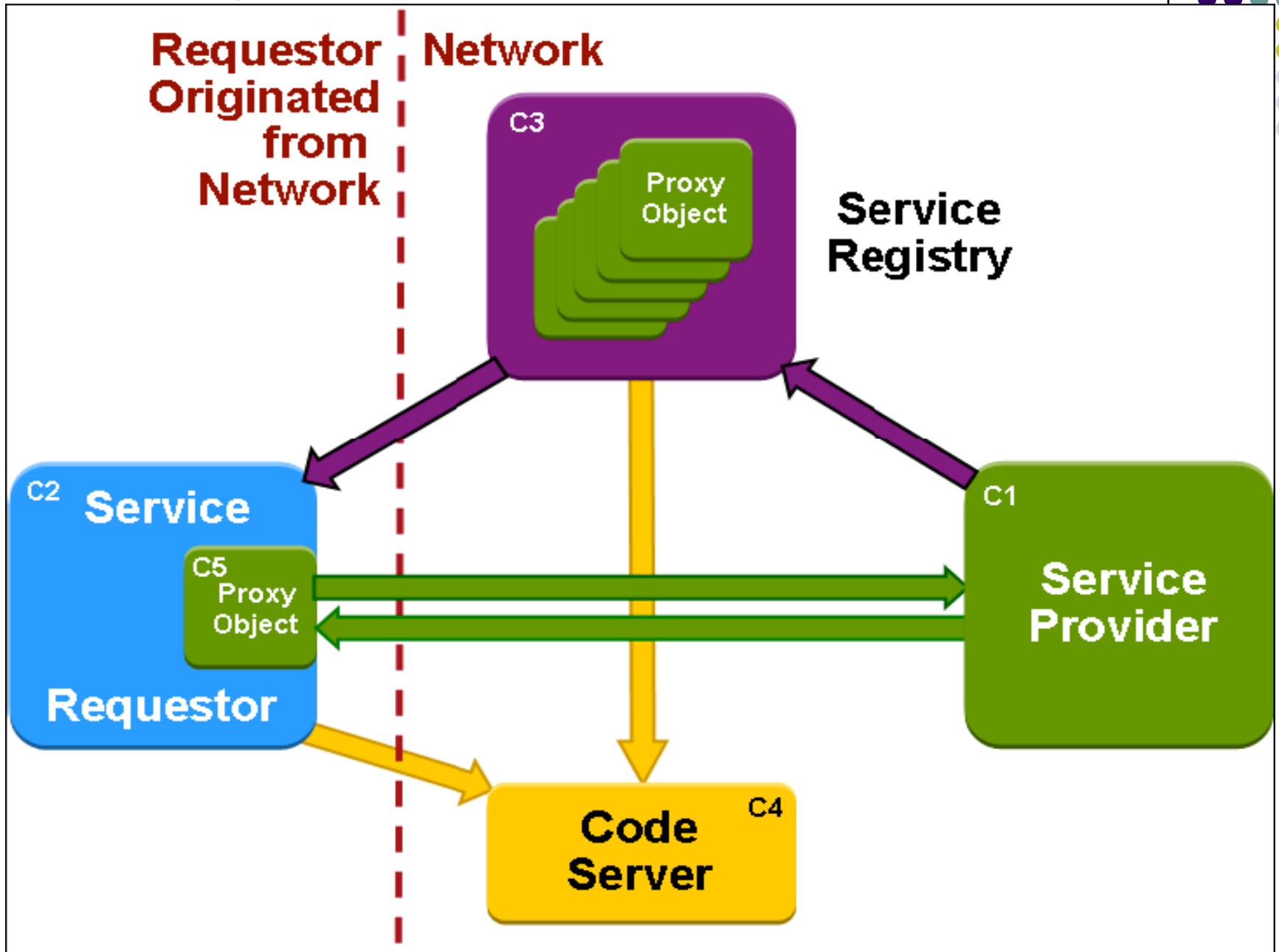
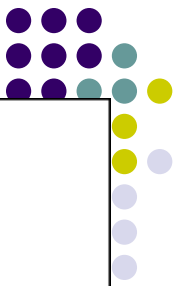


- **Sixth Generation RPC** – Federated Method Invocation allows for network invocations on multiple federated hosts.
- **Architectures to be noted** –
  - Service Oriented Architecture** – using loosely coupled software services that integrate them into distributed computing system by means of service oriented programming.
  - Federated Service Object Oriented Architecture** – Build on the object oriented distributed paradigm exemplified by the JINI architecture in which network objects come together on the fly to play their predefined roles.
  - Note** : SOOA differs from Service Protocol Oriented Architecture .

# Service Oriented Architecture



# Service Object Oriented Architecture



## Background : What is Service Context ?

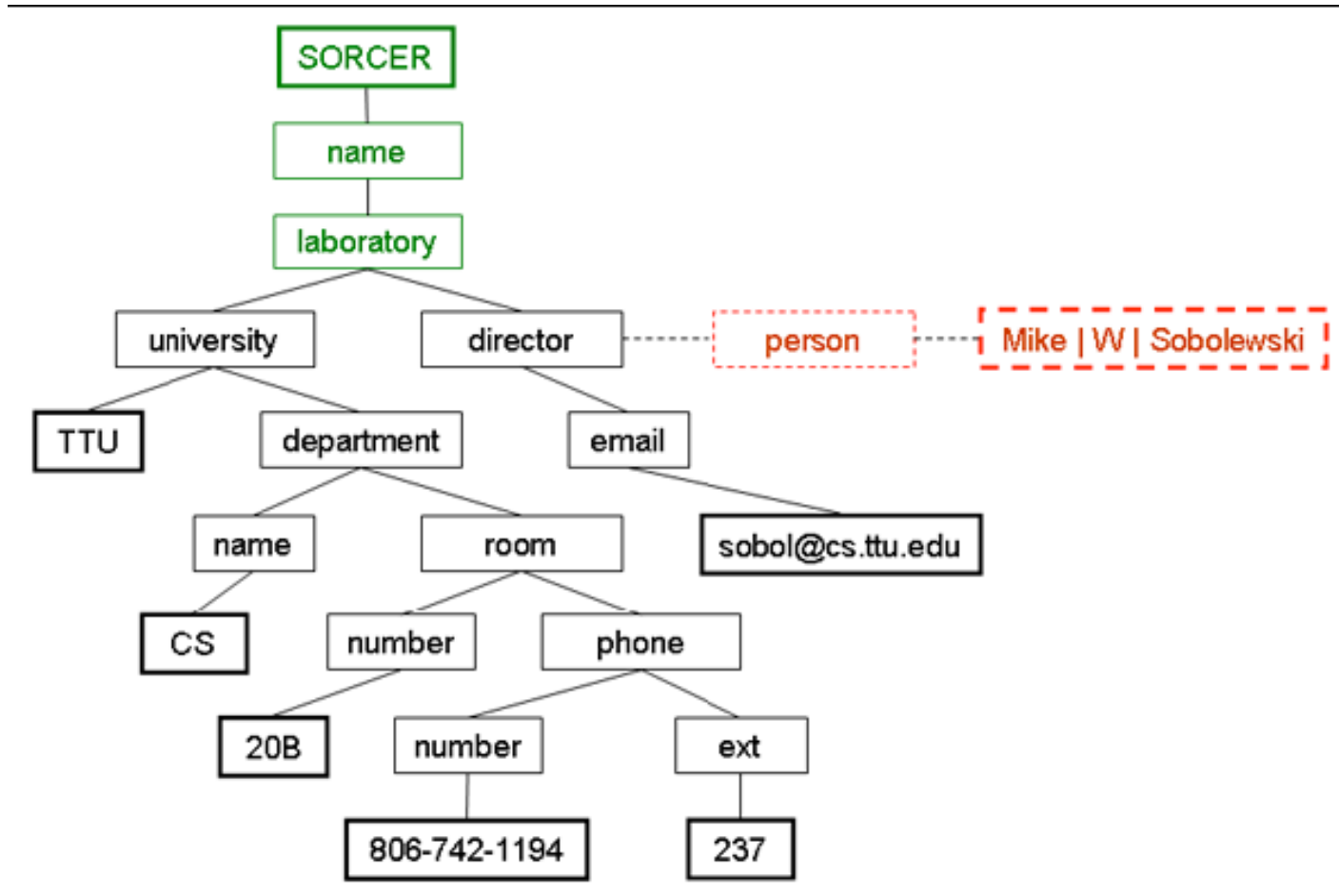
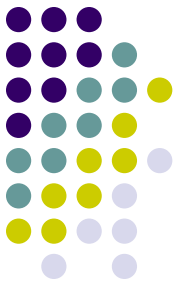
A service context, is a data structure that describes service provider ontology along with related data. It can be described conceptually as follows :



1. **context = [ subject ":" ] complement { complement }.**
2. **subject = element.**
3. **complement = element ";".**
4. **element = path [ "=" value ].**
5. **path = attribute { "/" attribute } [ { "<" association ">" } ] [ { "/" attribute }].**
6. **value = object.**
7. **attribute = identifier.**
8. **relation = domain product.**
9. **association = domain tuple.**
10. **product = attribute { "|" attribute }.**
11. **tuple = value { "|" value }.**
12. **attribute = identifier.**
13. **domain = identifier.**
14. **association = identifier.**
15. **identifier = letter { letter | digit }.**

# Background : What is Service Context ?

A service context, is a data structure that describes service provider ontology along with related data.



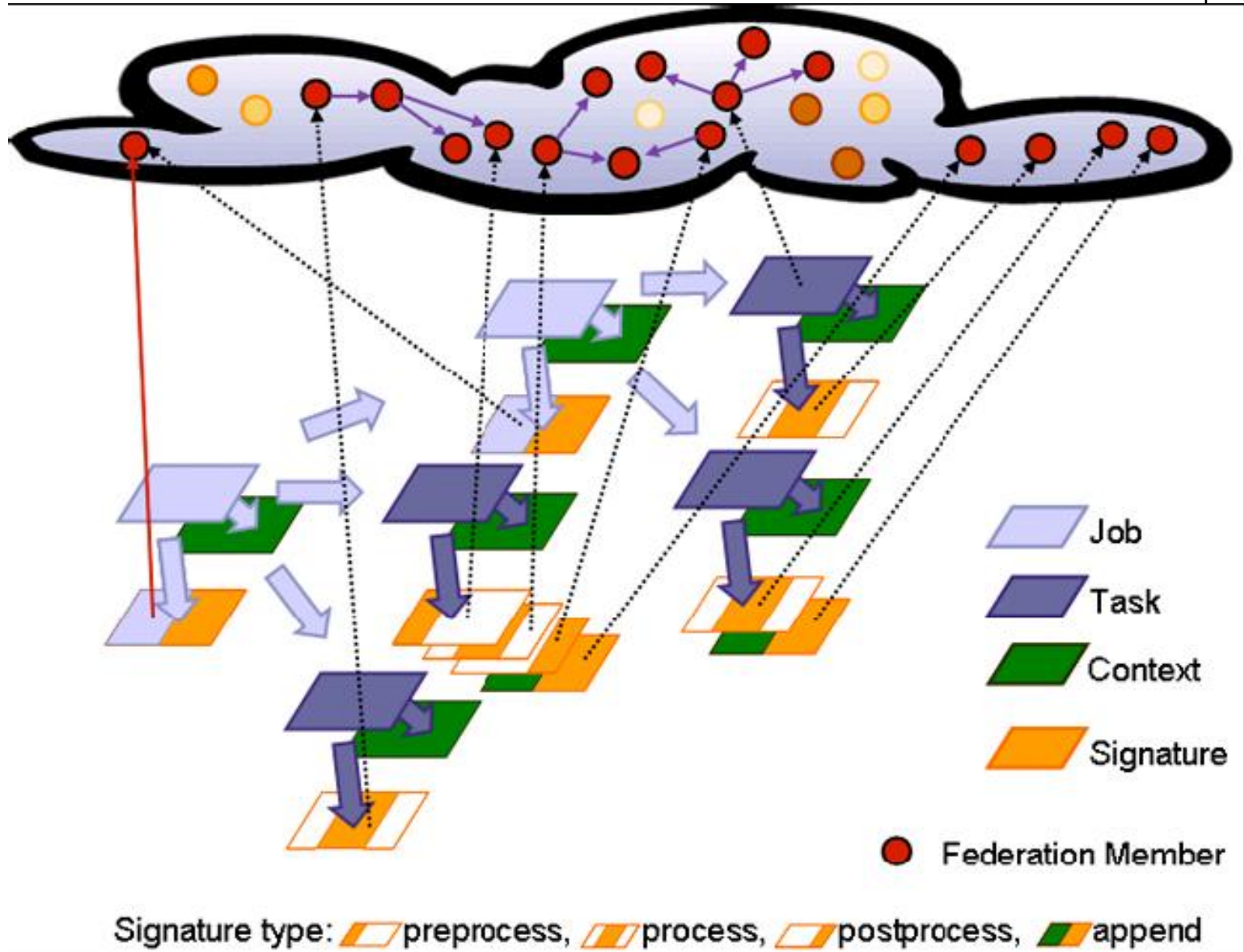
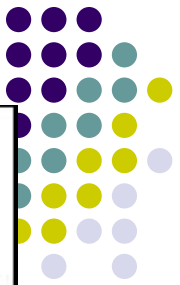
# Background : What is Service Context ?



Example of a Service Context :

- `laboratory/name = SORCER: university=TTU;`
- `university/department/name=CS;`
- `university/department/room/number=20B;`
- `university/department/room/phone/number=806-742-`
- `university/department/room/phone/ext=237;`
- `director <person | Mike | W | Sobolewski>`
- `/email=sobol@cs.ttu.edu;`
- `person | firstname | initial | lastname.`

# Background – Job Federation



### 3. Problem Statement



- The existing catalog service does not display sufficient data with respect to each context required for each remote method.
- As of today SORCER doesn't have a well defined framework for managing service contexts to support efficient interactive service-oriented programming.
- At present , we do not have a GUI , through which its possible to view, edit the Service Context.
- There is no well defined way to declare the Service Context and transport them for Service Interfaces .

## 4. Objective



**Develop a framework to display and manage contexts in **SORCER** with emphasis on modularity and low network usage.**

## 5. Methodology



- **Learn more about the inner workings of SORCER.**
- **Draw up a design for service context viewer, including use cases, component, interaction, and class diagrams.**
- **Design a graphical interface to display service contexts.**
- **Cataloger is the GUI which will be developed using the MVC framework.**

# UI Presentation

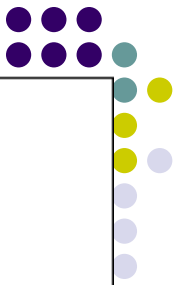
<b>Providers</b>	<b>Interfaces</b>	<b>Methods</b>	<b>Context</b>
<ul style="list-style-type: none"><li>•Provider 1</li><li>•Provider 2</li><li>•Provider 3</li></ul>			

<b>Providers</b>	<b>Interfaces</b>	<b>Methods</b>	<b>Context</b>
<ul style="list-style-type: none"><li>• <b><u>Provider 1</u></b></li><li>• Provider 2</li><li>• Provider 3</li></ul>	<ul style="list-style-type: none"><li>• Interface 1</li><li>• Interface 2</li><li>• Interface 3</li><li>• Interface 4</li></ul>		

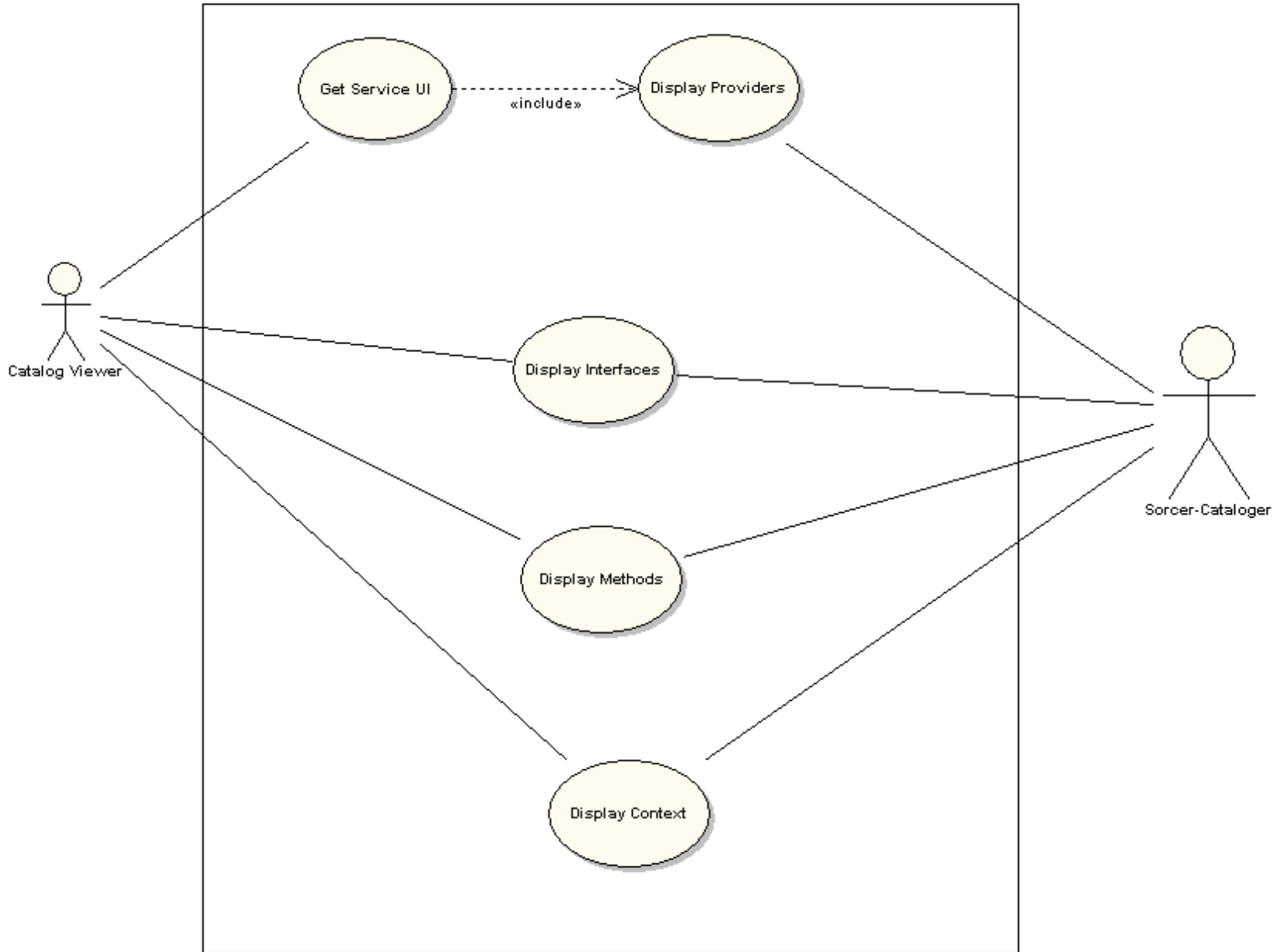
Providers	Interfaces	Methods	Context
<ul style="list-style-type: none"><li>• <b><u>Provider 1</u></b></li><li>• Provider 2</li><li>• Provider 3</li></ul>	<ul style="list-style-type: none"><li>• Interface 1</li><li>• <b><u>Interface 2</u></b></li><li>• Interface 3</li><li>• Interface 4</li></ul>	<ul style="list-style-type: none"><li>• Method 1</li><li>• Method 2</li><li>• Method 3</li><li>• Method 4</li><li>• Method 5</li></ul>	

Providers	Interfaces	Methods	Context
<ul style="list-style-type: none"><li>• <b><u>Provider 1</u></b></li><li>• Provider 2</li><li>• Provider 3</li></ul>	<ul style="list-style-type: none"><li>• Interface 1</li><li>• <b><u>Interface 2</u></b></li><li>• Interface 3</li><li>• Interface 4</li></ul>	<ul style="list-style-type: none"><li>• Method 1</li><li>• Method 2</li><li>• Method 3</li><li>• Method 4</li><li>• <b><u>Method 5</u></b></li></ul>	<ul style="list-style-type: none"><li>• Input Context</li><li>• Output Context</li></ul>

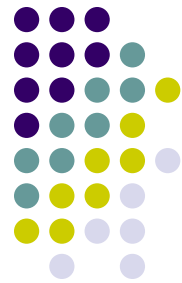
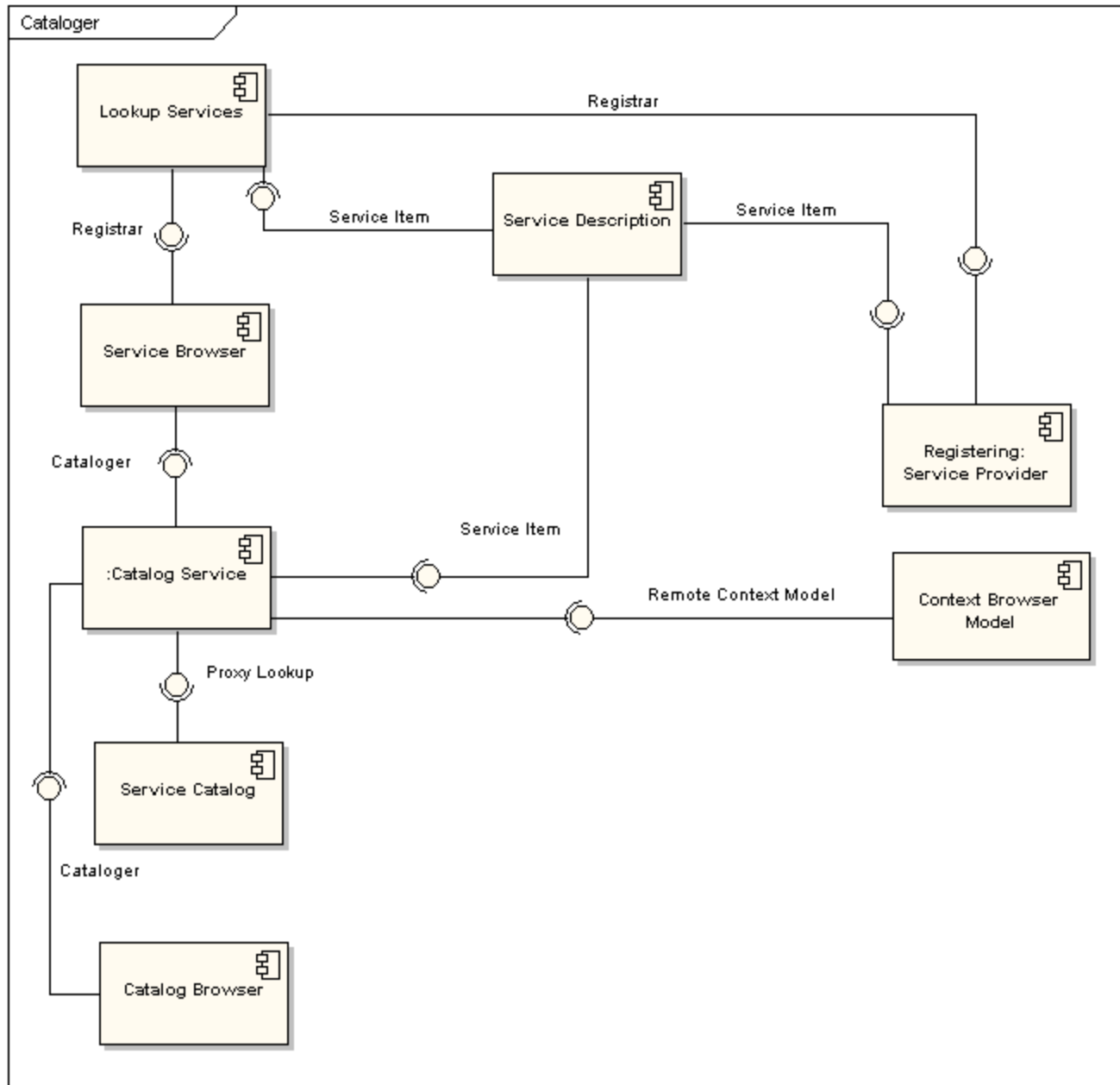
# Methodology - Use Cases



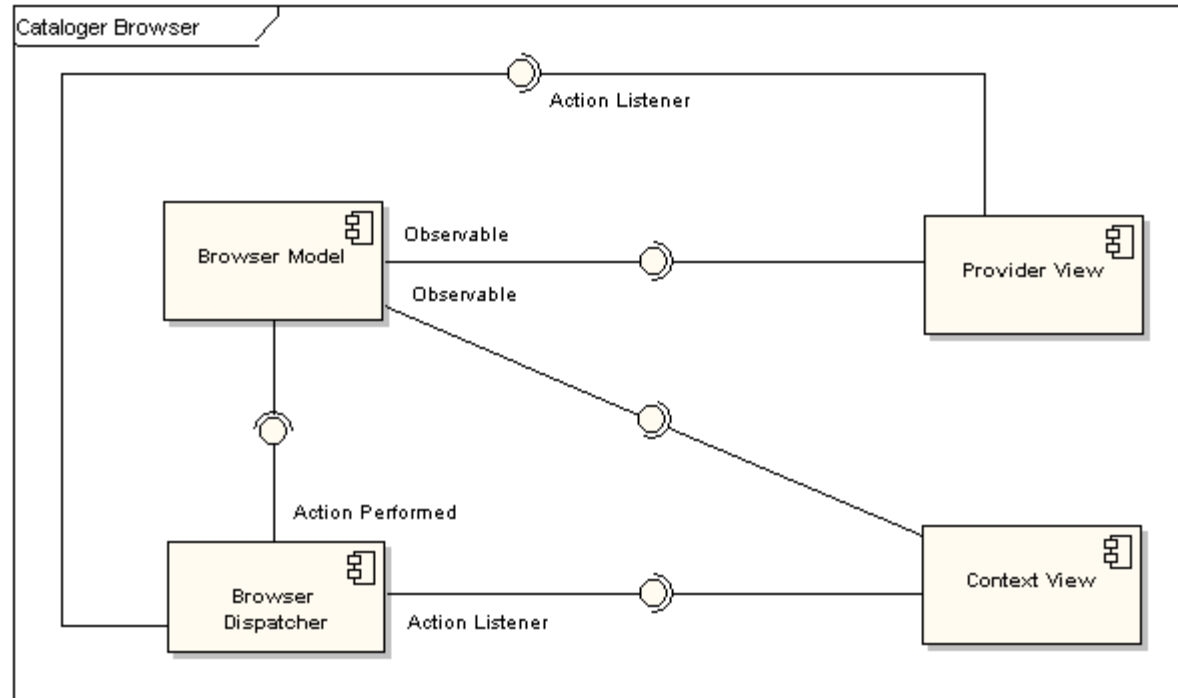
uc Use Case Model



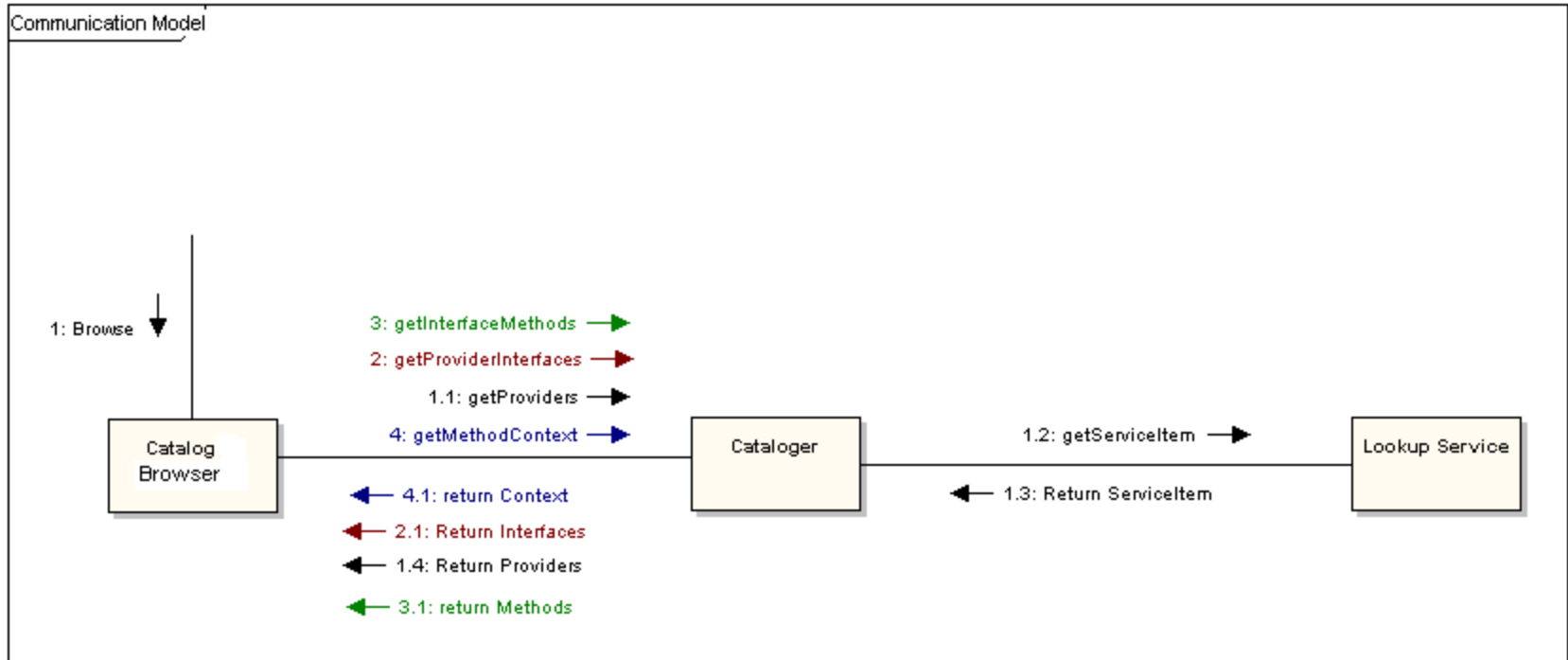
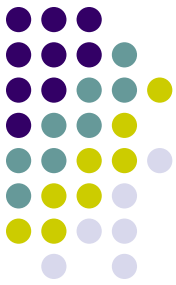
# Methodology – Component Diagram



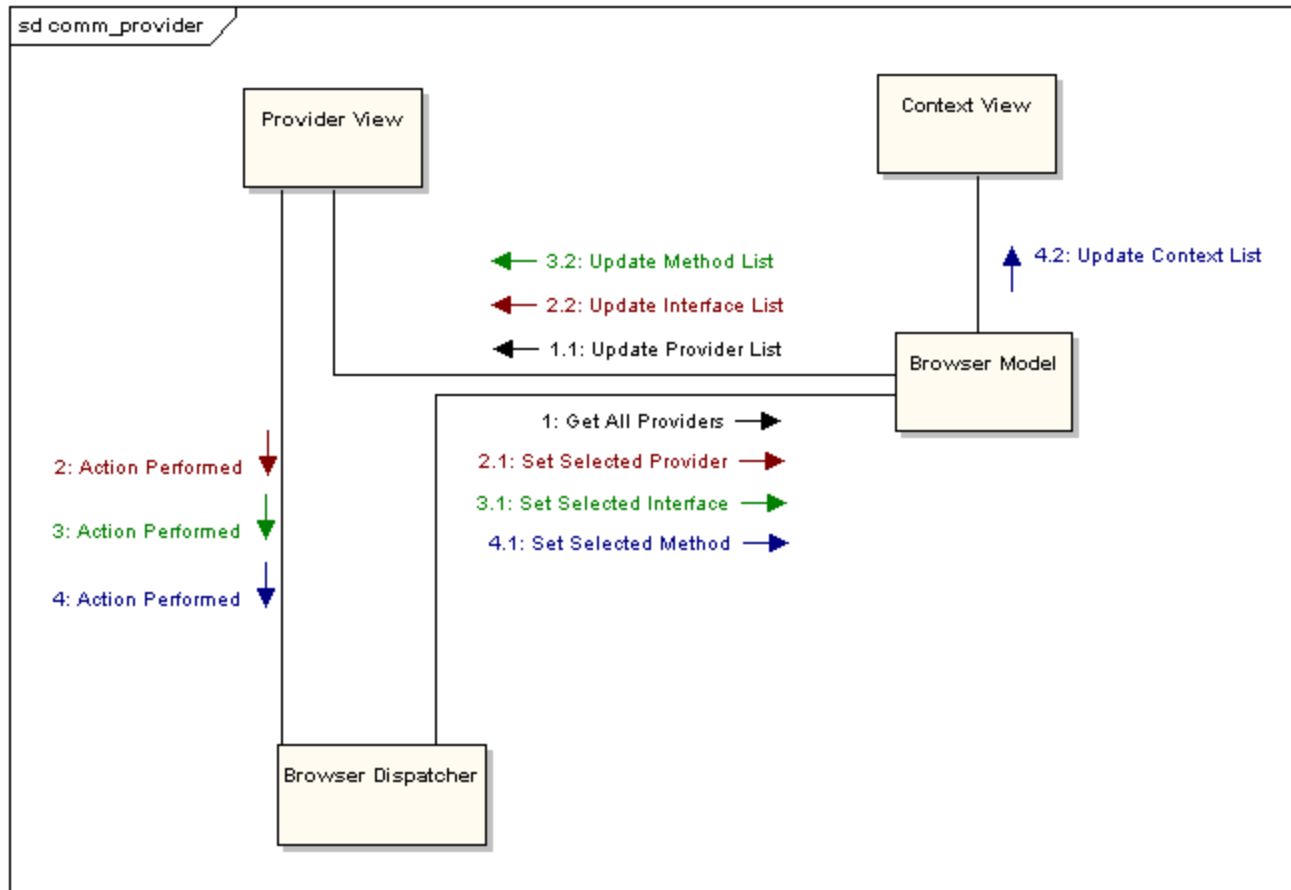
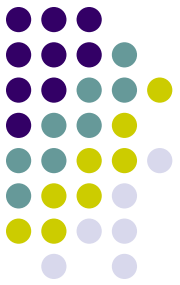
# Methodology – Component Diagram



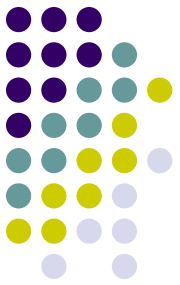
# Methodology – Interaction Diagrams



# Methodology – Interaction Diagrams



# Schedule



## Accomplished (03-30-07)

- **Literature review**

Review and analyze Jini Network Technology, JTree, SORCER;

- **Developed relevant Use Cases , Component diagrams , Interaction diagrams ;**

- **Able to display all the providers in the Catalog Browser .**

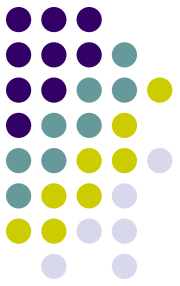
## Yet to be Completed :

- **Class Diagrams .**



# Benefits

- User friendly GUI , where the user can view all providers , interfaces , methods and the context .
- Standardized method context description .
- Help provide uniform and detailed service context access for interactive exertion-oriented programming .



# Q & A