Context-Aware Actors

Anne H. H. Ngu
Department of Computer Science
Texas State University-San Macos

Outline

- Why Context-Aware Actor?
- Context-Aware Scientific Workflow System
  - Architecture
  - Context Modelling and Provisioning
  - Context Providers and agents
  - Context Annotation
  - Demo of Context-Aware actors
  - Conclusion and future work

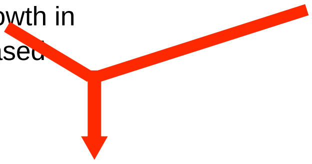
Why Context-Aware Actors?

- Actor-oriented scientific workflow model - ideal for modeling data intensive, stream-based and concurrent execution nature of scientific workflow.
- Problems with using Actor-oriented model for modeling dynamic workflows.
  - too many low-level control-flow actors and wiring lead to complex workflow that is hard to comprehend and re-use.
- Proposed solution in the past: Static Frame and Template, Dynamic frame actor, and Generic actor.
  - Static frame and template cannot adapt to runtime conditions (SciFlow 2006).
  - Dynamic frame encodes control-flow implementation of runtime conditions in the frame actor, thus cannot adapt to new situation without re-implementation (SSDBMS 2009).
  - Generic actor encodes all variations in control-flow logic a-priori in the actor.
  - All involve knowing low-level actor programming skills.
- Context-aware actor is proposed to enable actor-oriented scientific workflow to be more personalized, adaptive, intuitive, and intelligent by modeling the logic related to quality runtime adaptations as contexts and provisioned by a separate computing unit to the actor.

Context-Aware Actors

- **Context awareness**: the capability of being aware of its physical environment or situation (context) and responding proactively and intelligently based on such awareness
- One of the most important trends in computing that is becoming more important with relentless growth in mobile services (location-based services)

- **Actor**: a reusable component that may encapsulate external computation programs, grid services, scripts or local applications.



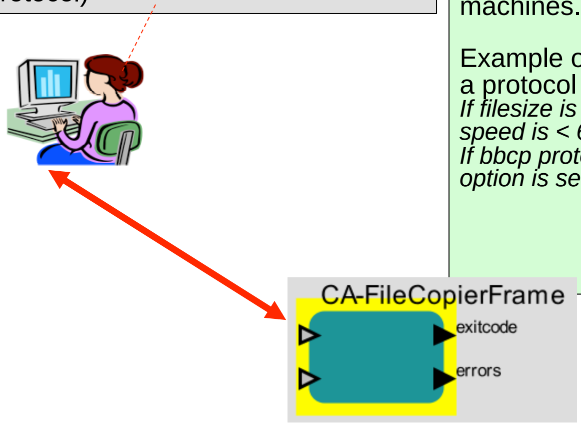
- **Context-aware Actors (CAA):** *make actors more personalized*, adaptive, and *intelligent*

Context-Aware FileCopier Actor
An Example CAA

User enters source file, destination file, source machine, destination machine, and a high-level file transfer option (e.g., fast protocol)

Recommend fastest type of protocol to transfer file taking user's contextual information such as size of the source file, availability of protocols and speed of the network connection between machines.

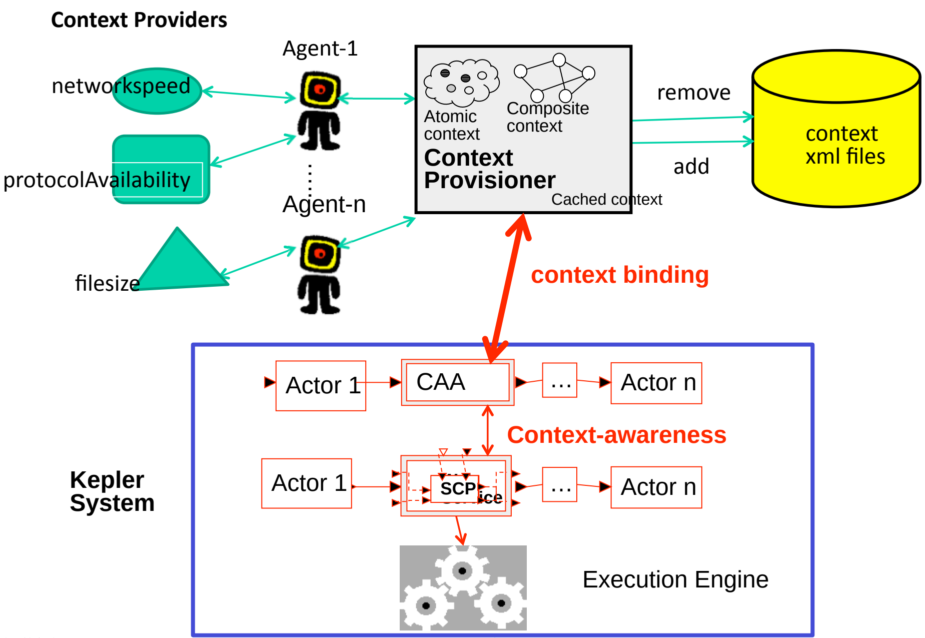
Example of contexts used for deciding on a protocol are:
If filesize is > 6GB and network connection speed is < 6ms, use bbcp protocol.
If bbcp protocol is not available and recursive option is set, use scp protocol



Context and Context-Awareness

- Context:
  - All data that can be gathered automatically at runtime which can affect the actor's behavior :
    - System parameters (OS, CPU usage, job queue status, network speed, type of machines, availability of resources and their versions, occurrences of certain events)
  - All data supplied by the user (gathered manually) especially those data related to his/her preferences. This can include strategies that can be applied to obtain specific level of guaranteed quality
- Context-Awareness
  - The mechanism for adapting the execution of an actor based on the sensed /gathered contextual information.

Kepler Context-Aware Scientific Workflow Architecture



Context Providers

networkspeed

protocolAvailability

filesize

Agent-1

Agent-n

Atomic context

Composite context

Context Provisioner

Cached context

remove

add

context xml files

context binding

Actor 1

CAA

…

Actor n

Context-awareness

Actor 1

SCP

ice

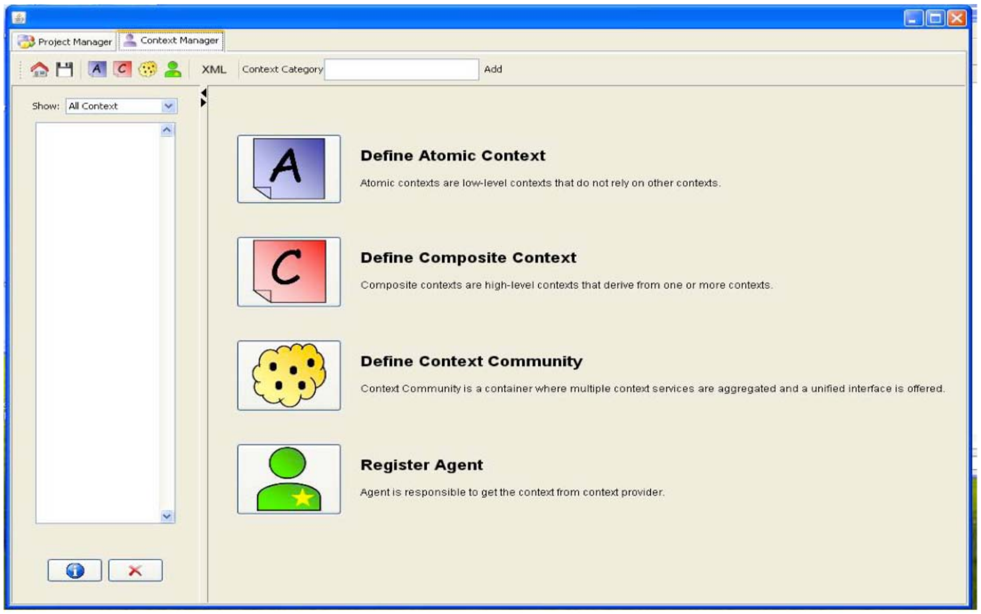
…

Actor n

Kepler System

Execution Engine

Context Provisioning Server



Define Atomic Context

Atomic contexts are low-level contexts that do not rely on other contexts.

Define Composite Context

Composite contexts are high-level contexts that derive from one or more contexts.

Define Context Community

Context Community is a container where multiple context services are aggregated and a unified interface is offered.

Register Agent

Agent is responsible to get the context from context provider.

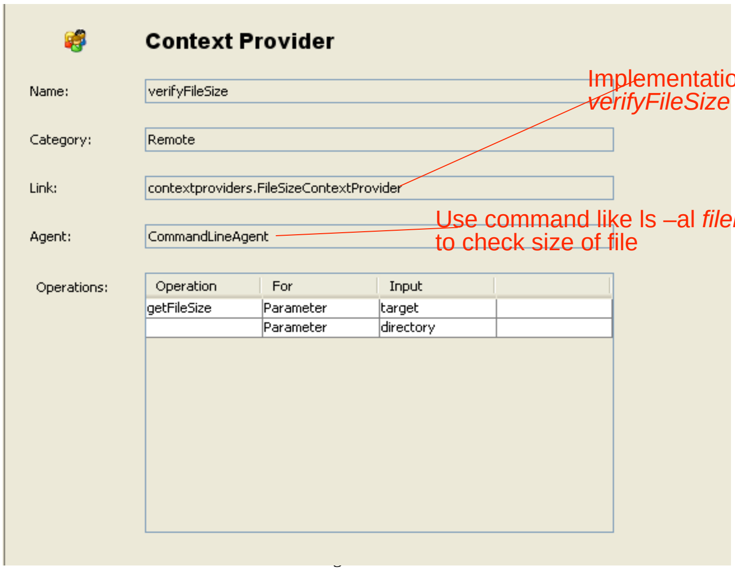
Context Provisioning Server (CP)

- Is adapted from the ContextServ project whose goal is to provide a platform for rapid development of context-aware web services. It is funded by Australian Research Council.
- CP is responsible for setting up, acquiring, and managing the contexts used by any context-aware actors.
- CP contains a set of agents and a repository of context XML files (context specification)
- The context XML files store the specification of all contexts used by a specific project. CP uses the context XML files to figure out how to acquire defined context information from the appropriate agents.
- CP is implemented as a web service using Apache CXF (a light-weight Java based web services development tool kits)

Context Providers

- Context providers are the context sources. They are independent from CP.
- There are many different kinds of context providers (hardware sensors, PDA, software systems, web services).
- Agents are used to provide a uniform abstraction for obtaining context from a particular kind of context provider (e.g. Webservice agent is used to acquire contexts from all context providers which can be queried through SOAP protocol)
- Context providers can be added and removed anytime.
- It is the responsibility of the context provider to provide the implementation of services (i.e. gathering of the contextual data).

An example of a context provider



Context Provider

Name: verifyFileSize

Category: Remote

Link: contextproviders.FileSizeContextProvider

Agent: CommandLineAgent

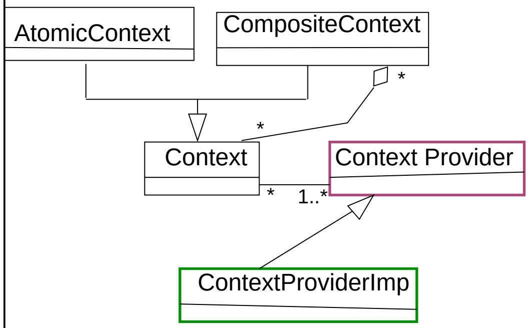
Operations:

Operation	For	Input
getFileSize	Parameter	target
	Parameter	directory

Implementation of *verifyFileSize* context provide

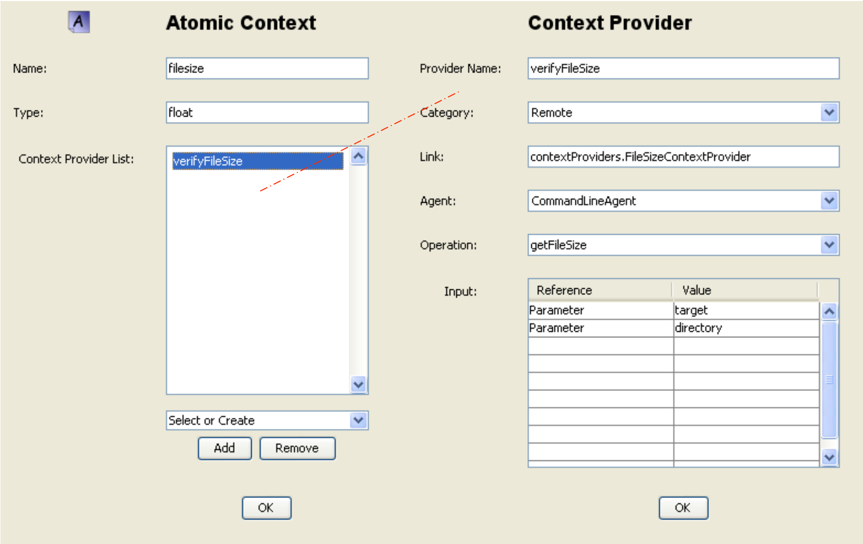
Use command like ls –al *filename* to check size of file

Context Modelling



- Context Provisioner distinguishes between *atomic* contexts and *composite* contexts
- **Atomic contexts**: low-level contexts, directly provided by context providers
  - e.g., filesize, temperature,
- **Composite contexts**: high-level contexts, no direct providers, aggregate multiple atomic or composite contexts
  - e.g., fastProtocol
  - Provide more powerful context modelling mechanism
- Context can be provided by one or more context providers

Example of Atomic Context



Atomic Context

Name: filesize

Type: float

Context Provider List: verifyFileSize

Context Provider

Provider Name: verifyFileSize

Category: Remote

Link: contextProviders.FileSizeContextProvider

Agent: CommandLineAgent

Operation: getFileSize

Input:

Reference	Value
Parameter	target
Parameter	directory

XML representation of atomic context

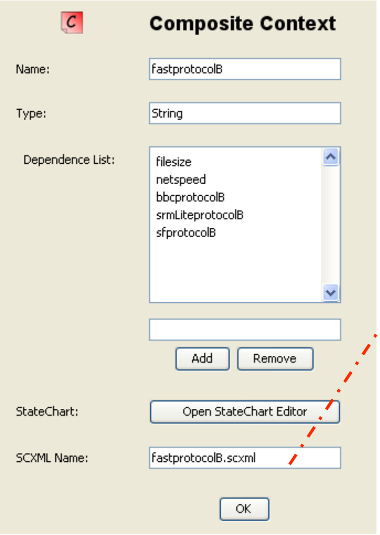
```
<context>
    <name>filesize</name>
    <type>float</type>
    <category>Atomic</category>
    <context_provider>
         <name>verifyFileSize</name>
          <category> Remote</category>
          <agent>commandLineAgent>
          <link>contextProviders.FileSizeContextprovider</link>
           <operation>getFileSize</operation>
           <input ref=“Parameter”> target</input>
           <input ref=“Parameter”>directory</input>
        </context_provider>
 </context>
```

Rules for recommending a fast protocol defined as a composite context

Fast protocol:

```
If (  (bbcp is available)
    // big file, slow network speed, so better off with a fast protocol
    if (filesize is >=6 GB) &   (network speed > =6ms) )  then  choose bbcp
    else if (filesize < 6 GB  OR networkspeed < 6 ms )
         if recursive transfer
           choose scp
         else
           if (sftp available) choose sftp
           else choose scp
 else
       if recursive transfer
               choose scp
      else
           if  (sftp available)  choose sftp
           else choose scp
```

Example of Composite Context



Composite Context

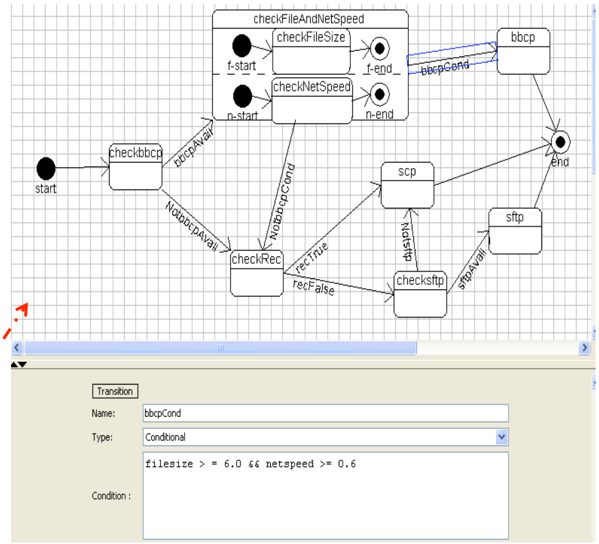
Name: fastprotocolB

Type: String

Dependence List: filesize, netspeed, bbcpprotocolB, srmLiteprotocolB, sfprotocolB

StateChart: Open StateChart Editor

SCXML Name: fastprotocolB.scxml



Transition

Name: bbcpCond

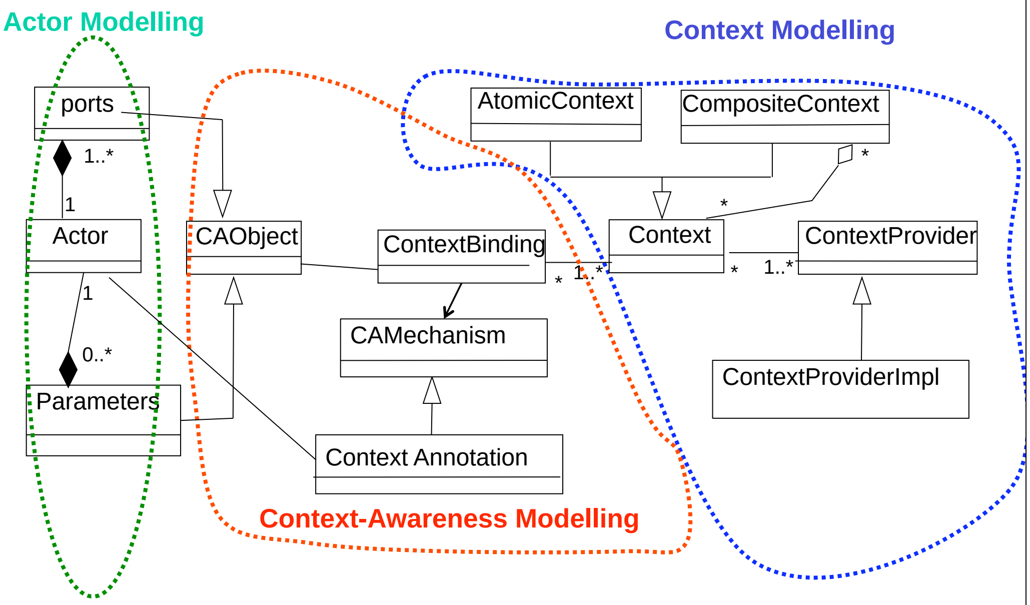
Type: Conditional

Condition : `filesize > = 6.0 && netspeed >= 0.6`

Representation of composite context

```
<context>
     <name>fastprotocolB</name>
     <type>String</type>
     <category>Composite</category>
     <dependence>filesize</dependence>
     <dependence>netspeed</dependence>
     <dependence>bbcprotocolB</dependence>
     <dependence>srmLiteprotocolB</dependence>
      <dependence>sfprotocolB</dependence>
      <scxml>fastprotocolB.scxml</scxml>
 </context>
```

Development of Context-Aware Actor



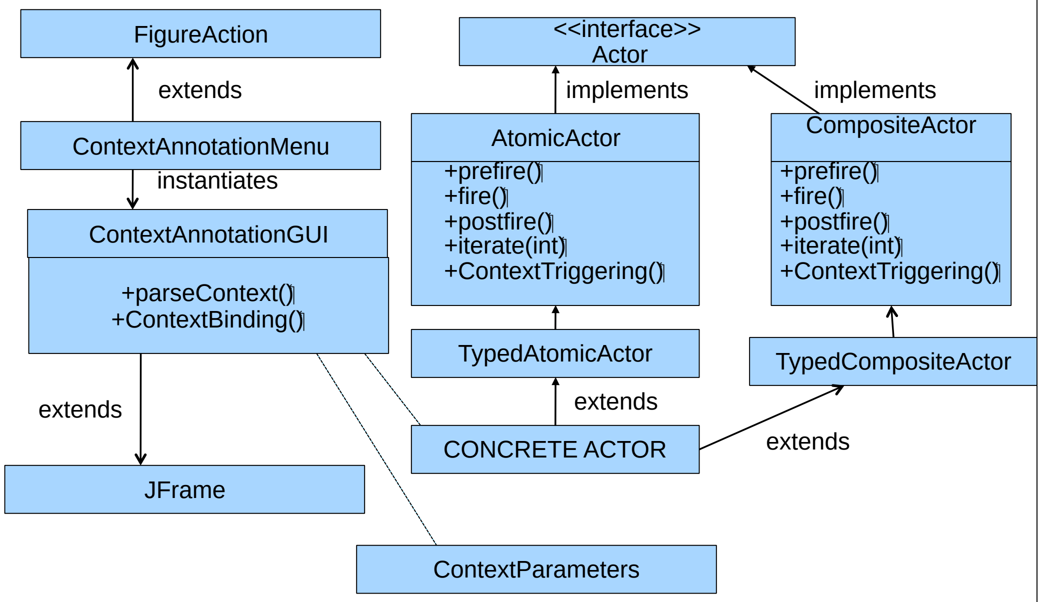
Context Awareness Modeling

- Context binding is the ability for an actor to bind to/access relevant contexts defined in the CP.
- Context awareness is the mechanism by which an actor behavior can be modified based on the gathered contextual information.
- A context-aware scientific workflow system must implement a context binding mechanism and a context-triggering mechanism.

Context Annotation as the Awareness Mechanism

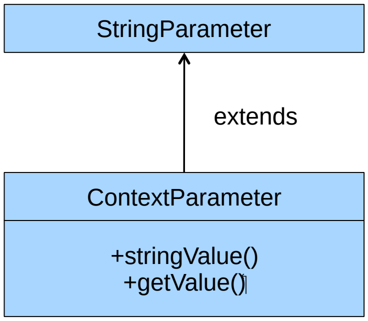
- The main purpose of context annotation is to facilitate an end user (scientist) to tailor an existing actor in the KEPLER repository to his/her environmental context by interacting with a visual annotation tool. Context is injected into an existing workflow on demand –basis.
- The end user must identify context sensitive parameters (CAObject) in an actor and bind them to appropriate contexts. We assume that the end user understand the semantics of the defined contexts in his/her domain.
- Two new classes, ContextAnnotationGUI and ContextParameter are introduced. Context annotation encapsulates all user interactions with the context provisioning server that will result in correct context binding.
- ContextParameter class encapsulates actions associated with contacting the context provisioning system to provide real-time evaluation of the contextual value of a context sensitive port or parameter in an actor.
- Context Annotation provides a generic mechanism for incorporating context without any code changes.

Implementation of Context Annotation



FigureAction
extends
ContextAnnotationMenu
instantiates
ContextAnnotationGUI
+parseContext()
+ContextBinding()
extends
JFrame
<<interface>>
Actor
implements
AtomicActor
+prefire()
+fire()
+postfire()
+iterate(int)
+ContextTriggering()
implements
CompositeActor
+prefire()
+fire()
+postfire()
+iterate(int)
+ContextTriggering()
TypedAtomicActor
TypedCompositeActor
extends
CONCRETE ACTOR
extends
ContextParameters

Implementation of ContextParameter



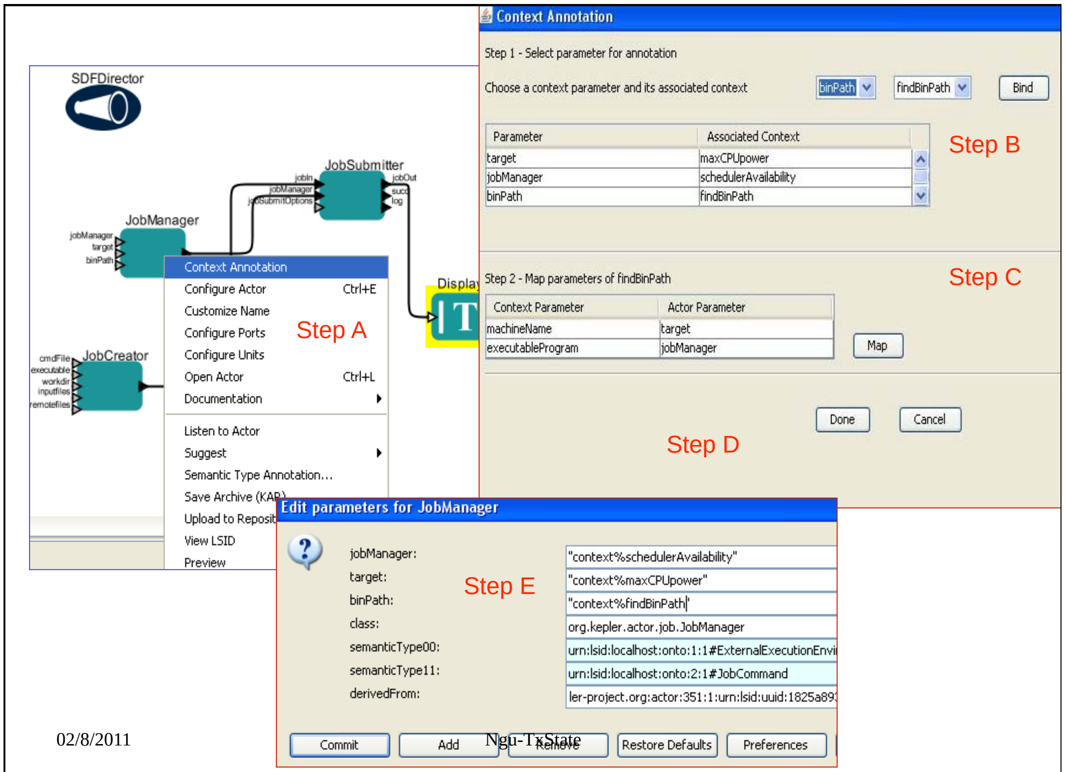
StringParameter
extends
ContextParameter
+stringValue()
+getValue()

```
public String getValue(Object[][] paras) throws
IllegalActionException
{

// contacts the Context Provisioning server
//saves the list of parameters from an actor
//uses getContext() API to evaluate the context
```

Steps in Context Annotation

1. From the GUI, display all the existing parameters or ports of the actor that are candidates for context annotation.
2. Let the user selects the context parameter for annotation.
3. Contacts the CP server to find the list of contexts that is relevant to the selected context parameter.
4. Let the user picks the required context to bind to.
5. If the binding requires mapping, map the context variables with actor parameters.
6. Click “map” to finish the binding and repeat 2 to 5 for the next parameter.
7. The context triggering phase started when user pressed the done button and run the actor.
8. The context computation takes place during the firing of the actor



Step 1 - Select parameter for annotation

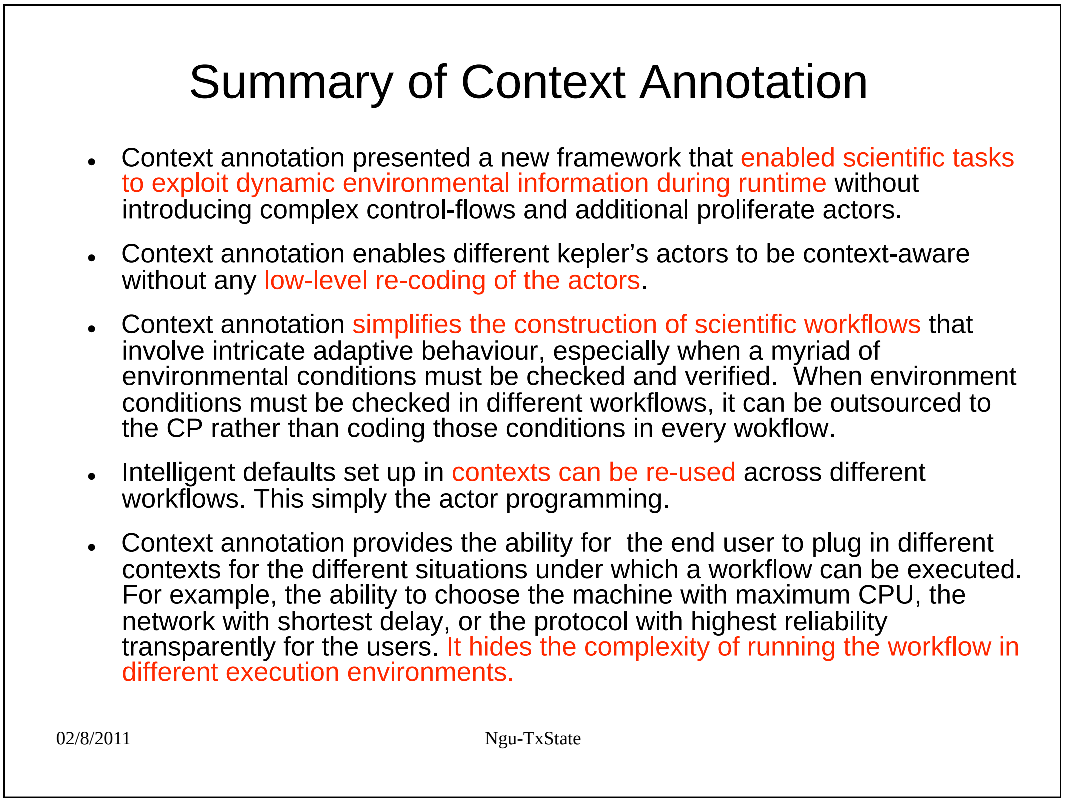
Choose a context parameter and its associated context: binPath, findBinPath

Parameter	Associated Context
target	maxCPUpower
jobManager	schedulerAvailability
binPath	findBinPath

Step 2 - Map parameters of findBinPath

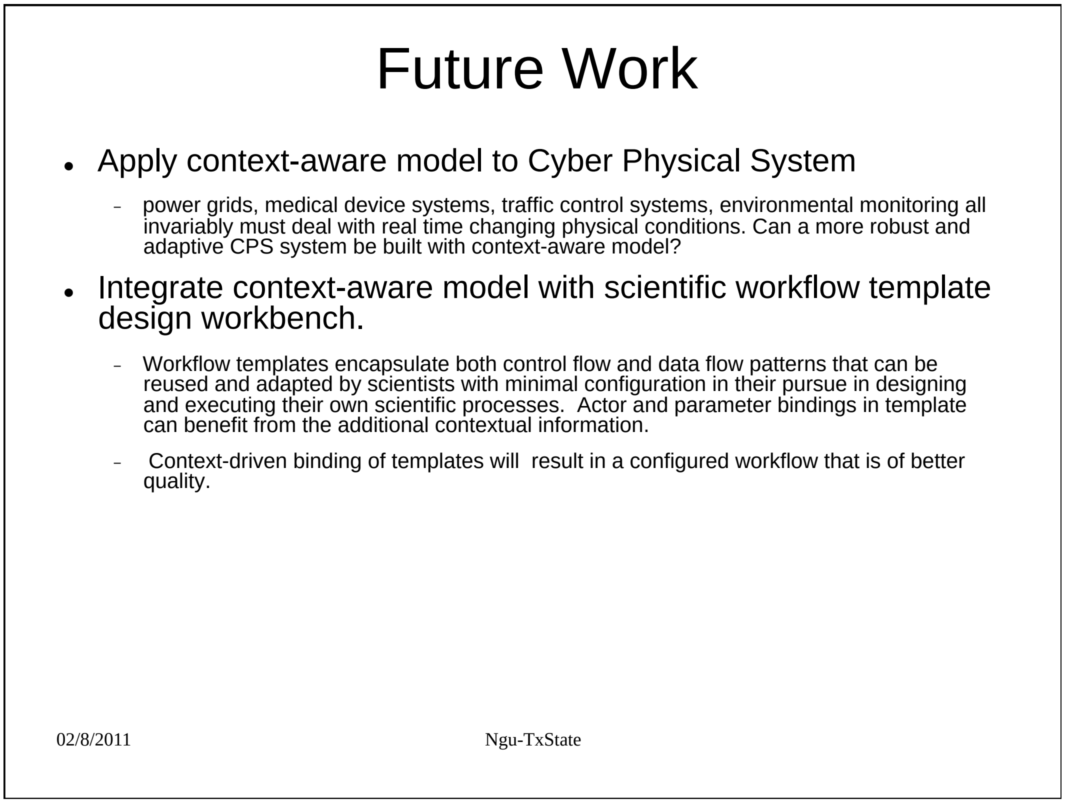
Context Parameter	Actor Parameter
machineName	target
executableProgram	jobManager

jobManager:	"context%schedulerAvailability"
target:	"context%maxCPUpower"
binPath:	"context%findBinPath"
class:	org.kepler.actor.job.JobManager
semanticType00:	urn:lsid:localhost:onto:1:1#ExternalExecutionEnvi
semanticType11:	urn:lsid:localhost:onto:2:1#JobCommand
derivedFrom:	ler-project.org:actor:351:1:urn:lsid:uuid:1825a89



Summary of Context Annotation

- Context annotation presented a new framework that enabled scientific tasks to exploit dynamic environmental information during runtime without introducing complex control-flows and additional proliferate actors.
- Context annotation enables different kepler's actors to be context-aware without any low-level re-coding of the actors.
- Context annotation simplifies the construction of scientific workflows that involve intricate adaptive behaviour, especially when a myriad of environmental conditions must be checked and verified. When environment conditions must be checked in different workflows, it can be outsourced to the CP rather than coding those conditions in every wokflow.
- Intelligent defaults set up in contexts can be re-used across different workflows. This simply the actor programming.
- Context annotation provides the ability for the end user to plug in different contexts for the different situations under which a workflow can be executed. For example, the ability to choose the machine with maximum CPU, the network with shortest delay, or the protocol with highest reliability transparently for the users. It hides the complexity of running the workflow in different execution environments.



Future Work

- Apply context-aware model to Cyber Physical System
  - power grids, medical device systems, traffic control systems, environmental monitoring all invariably must deal with real time changing physical conditions. Can a more robust and adaptive CPS system be built with context-aware model?
- Integrate context-aware model with scientific workflow template design workbench.
  - Workflow templates encapsulate both control flow and data flow patterns that can be reused and adapted by scientists with minimal configuration in their pursue in designing and executing their own scientific processes. Actor and parameter bindings in template can benefit from the additional contextual information.
  - Context-driven binding of templates will result in a configured workflow that is of better quality.