



Task-Centered Performance Analysis and Modeling

Fatma Mili, Swapna Ghanekar

Oakland University

Sanjiv Dungrani, TARDEC

5th Annual Intelligent Vehicle Systems
Symposium and Exhibition

June 13-16, 2005, Traverse City

http://register.ndia.org/interview/register.ndia?PID=Brochure&SID=_11O0IK27A&MID=C455



Introduction, Motivation



System design must account for the role of human operators. Relevant questions include:

- What tasks should be assigned to operators?
- What combinations of tasks should be assigned to the same operator?
- What tasks are better automated?
- What support can the system provide to human operators?



Traditional Approach



1. Create a Design with built-in role for human operators.
2. Repeat
 1. Develop performance models.
 2. Analyze performance models.
 3. Revise the design as neededUntil design is satisfactory



Shortcomings of this traditional approach



- High up front investment.
- Too tedious and inflexible.
- Analysis and evaluation comes too late.
- Only one type (grain size) of analysis (minute details or nothing).
- Little reuse



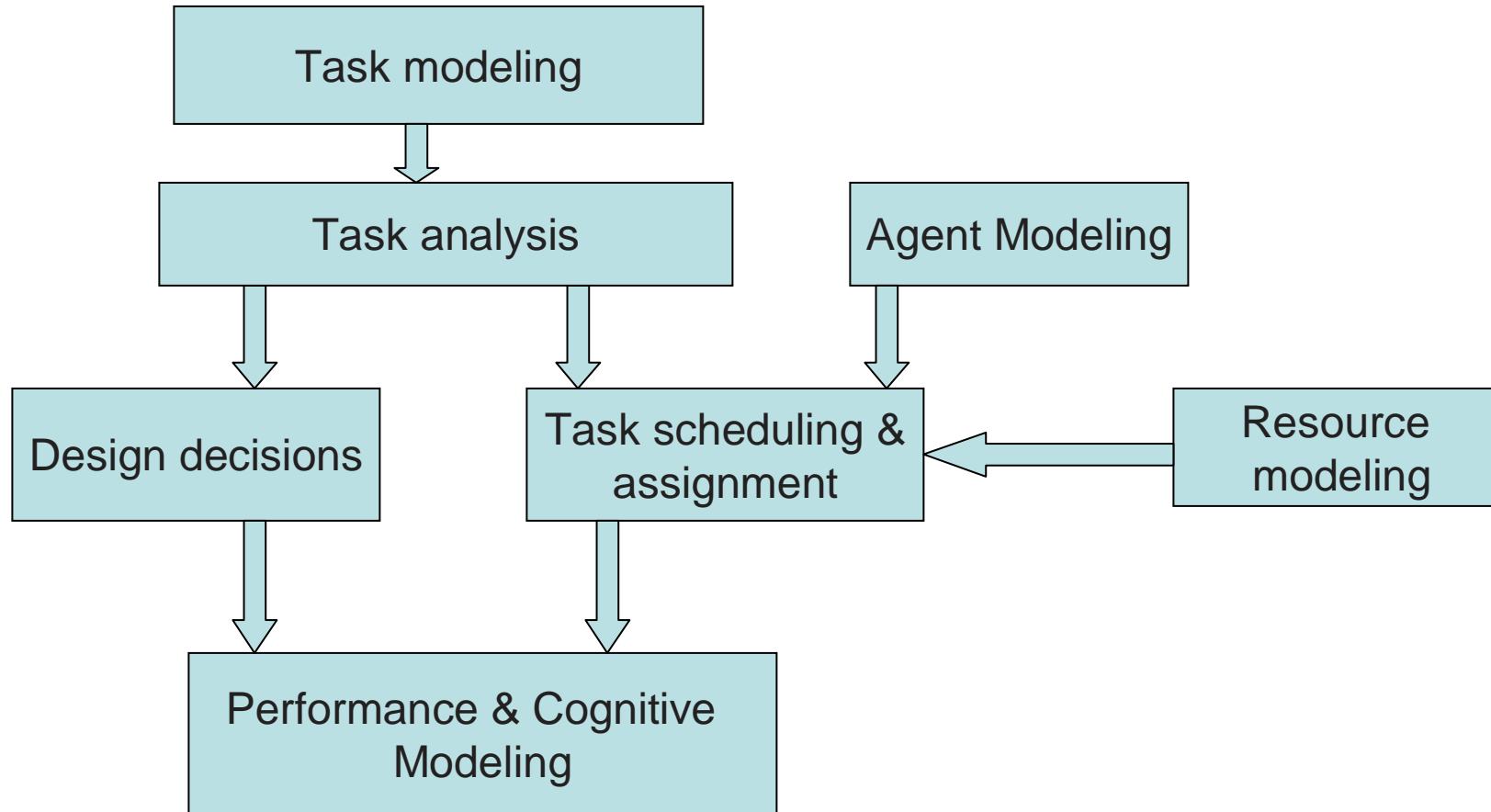
Little reuse



- Performance models developed from scratch.
- Models developed for one configuration cannot be reused for another configuration
- Information collected about one task assigned to one agent cannot be reused if the task is assigned to different agent



Solution, Approach



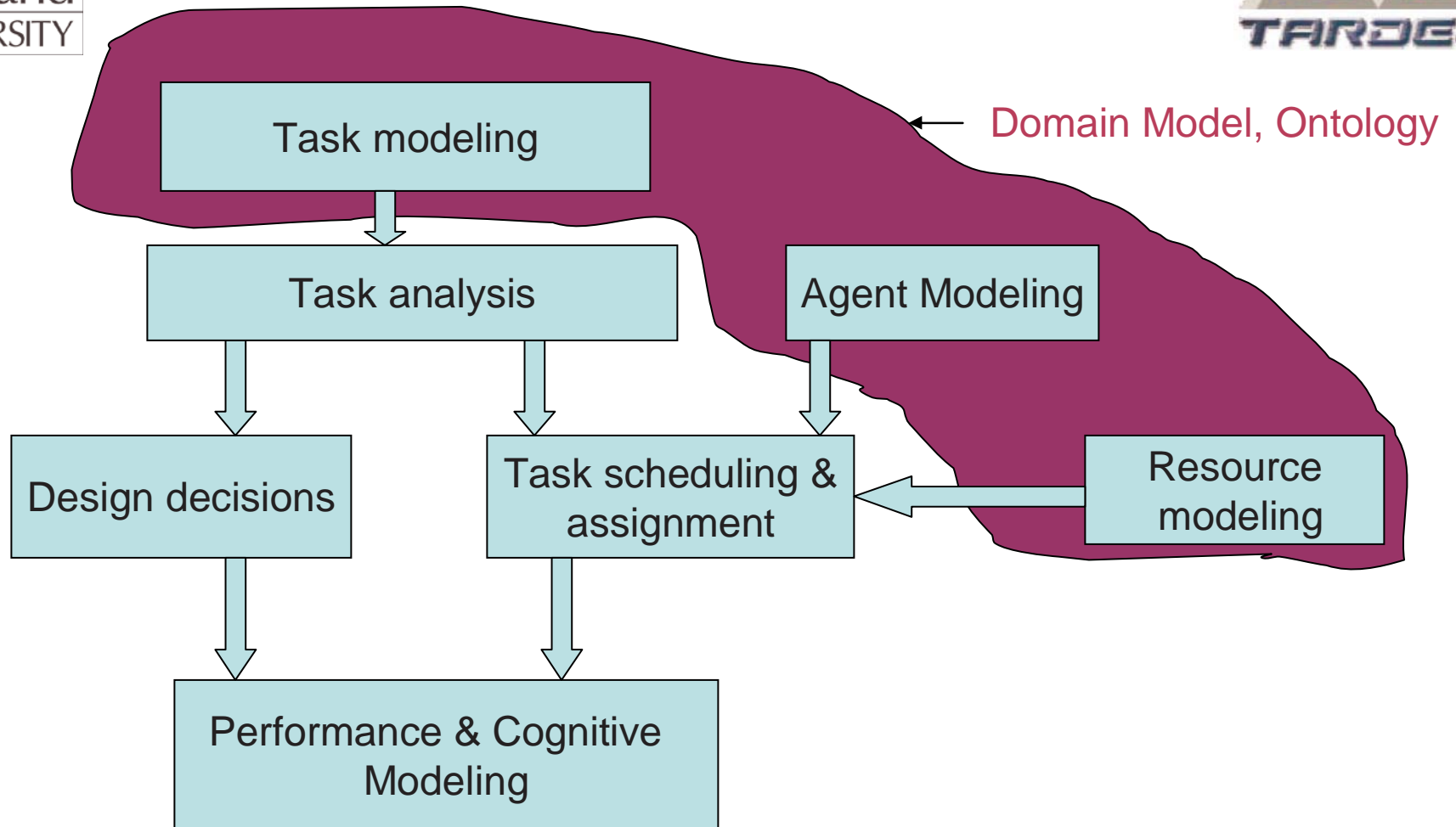


Task Modeling

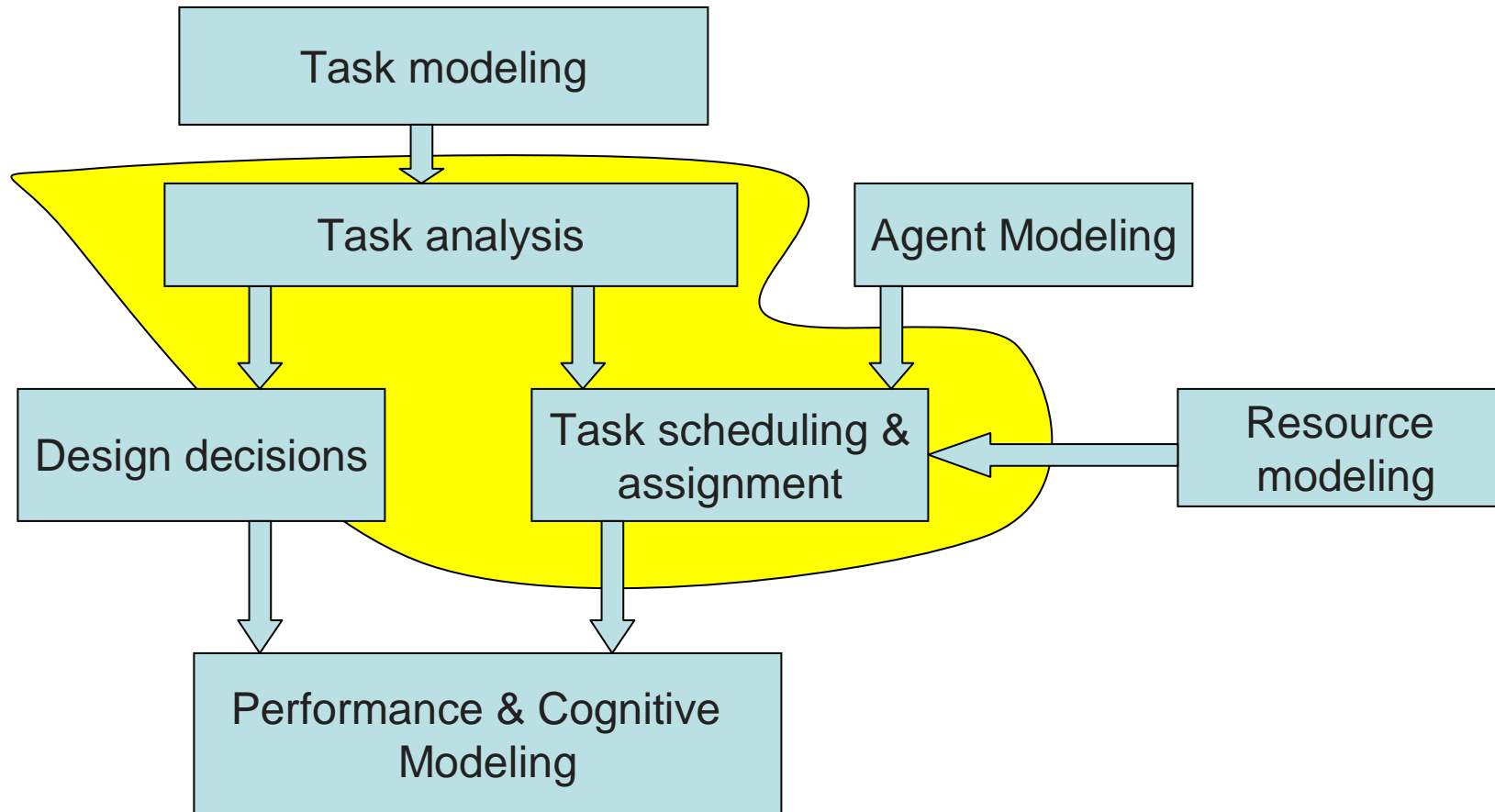


- ❑ Develop a repository (ontology) of tasks.
 - Repository must contain all information inherent to the task relevant to performance.
 - Repository must contain no information specific to agents performing the task
- ❑ Ontology becomes an evolving reflection of our understanding of tasks.
- ❑ Ontology used to develop alternative designs, reason about them, and compare them.

Scope of the Ontology



Insight gained from the ontology





Premise of the approach



- Task knowledge is relatively stable
- Task knowledge reuse is important.
- Ontologies are a good vehicle for capturing this knowledge.
- Ontologies are a good vehicle for sharing this knowledge.
- Ontologies are a good vehicle for reasoning about this knowledge.
- Ontologies are a good vehicle for evolving this knowledge.



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models.
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Reusing Vocabularies and Ontologies
 3. Tools for constructing and maintaining domain model.
4. Summary Conclusion



1.1 Motivation



1. Gap in grain size
2. Separation of concerns
3. Reusability of modeling information
4. Early insight about tasks, task assignments, and system design



Motivation: Gap in grain size



System level missions	Performance modeling tasks
Conducting a reconnaissance march.	Turning the steering wheel
Directing air traffic around the airport (see example 1)	Selecting new trajectory direction and speed using the mouse(see example 2)



Example 1: CE-6



In response to growth in air traffic, FAA responded with revised decentralized procedures for Air Traffic Management (ATM).

New procedure covers 15 Concept Elements CE-0 through CE-14.

CE-6: En Route Trajectory Negotiation.

See references 1, 2.



Example 1 cont': Concept Element -6



- ❑ Objectives [2]:
 - Negotiate safe routes
 - Reduce un-necessary route deviations
 - Reduce workload for AT Service Providers
 - Facilitate change requests
 - Account for “user” preferences.
- ❑ New CE needs to be tested through modeling
- ❑ Skeletal Description of CE-6 next slide taken from [3].

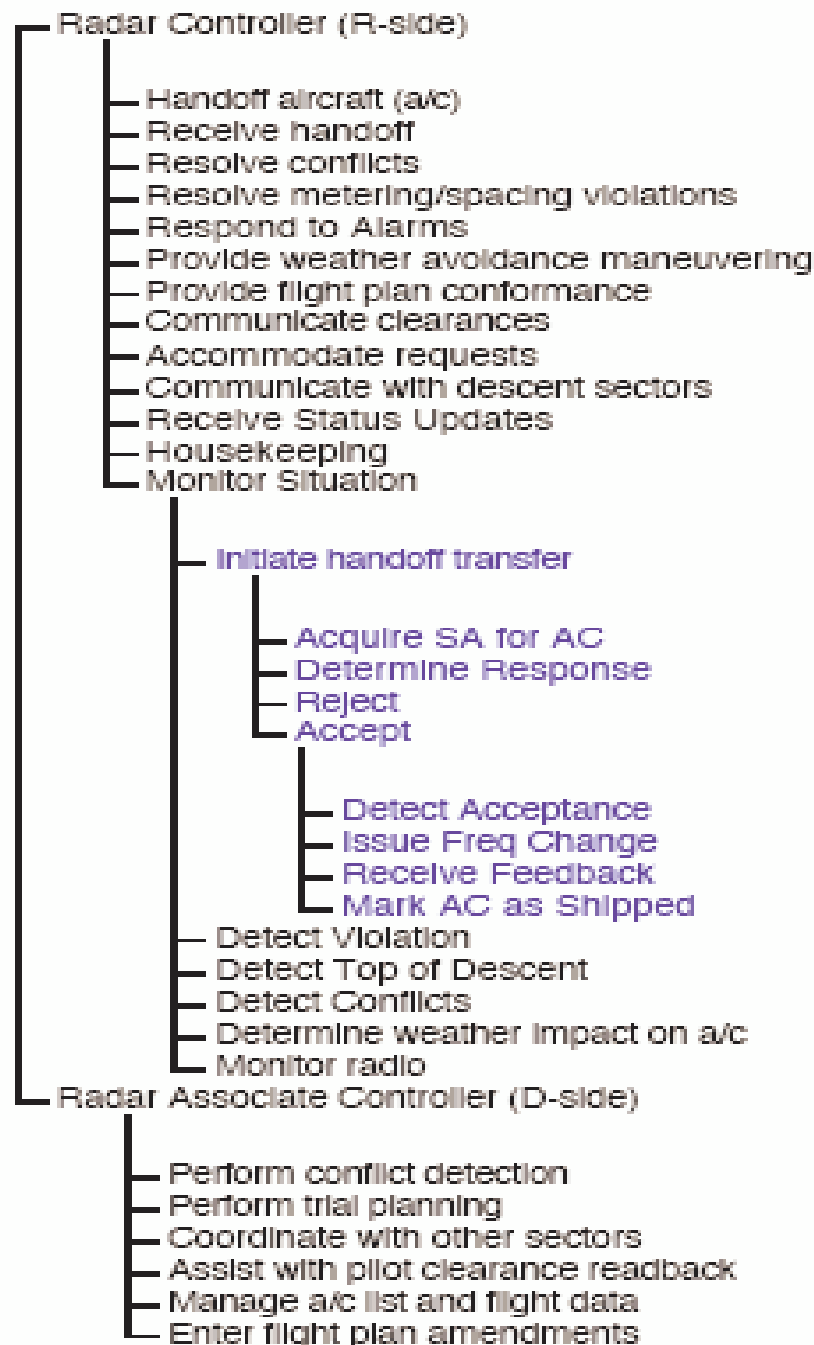
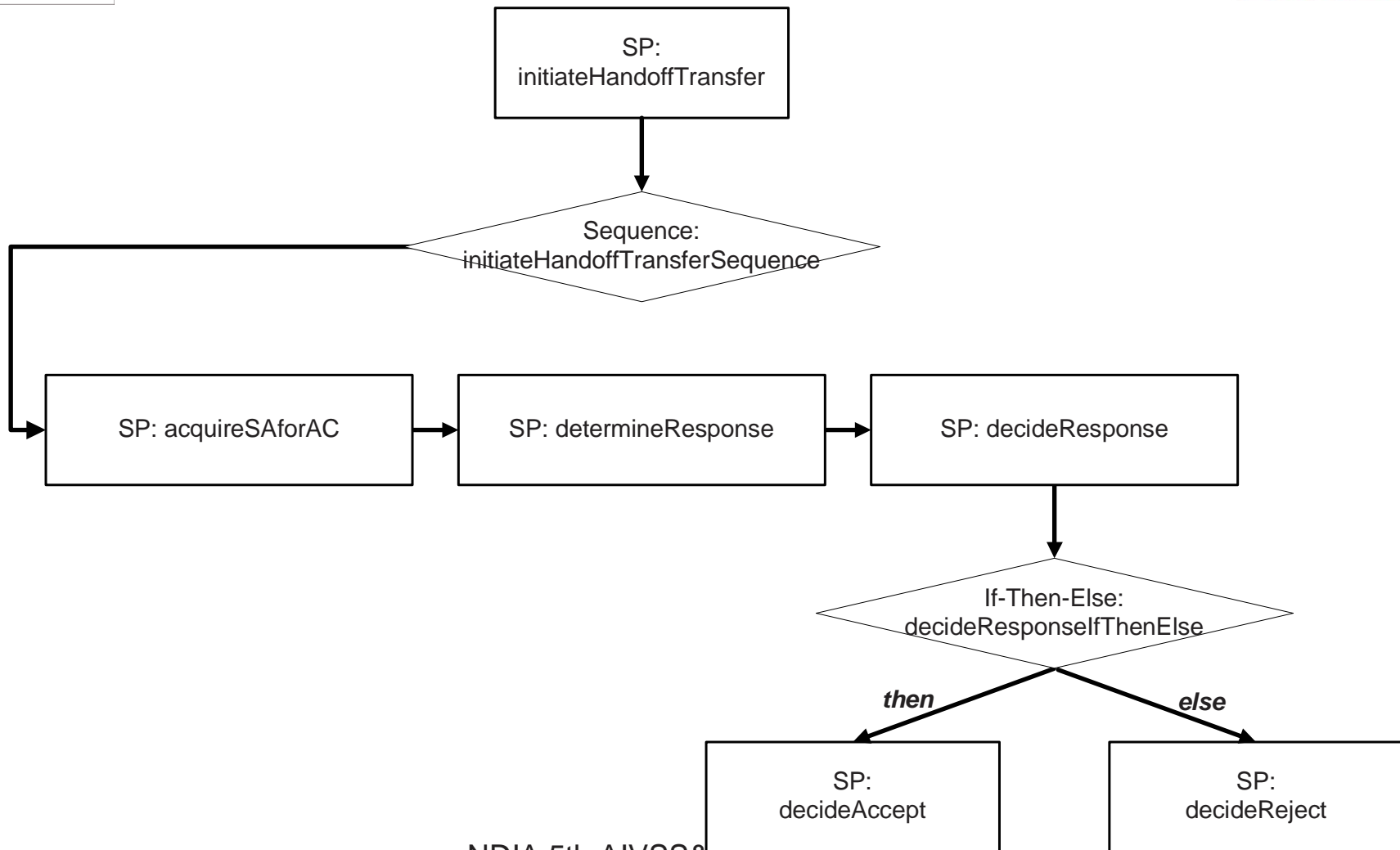


Figure 4: Functional analysis of enroute control



Example 1: Decomposition





Example 2: Selection



- Select a new trajectory (direction and speed) using mouse
- Steps include
 - Finding the correct item on the screen
 - Moving the mouse to the item
 - Verifying that the item is correct
 - Selecting the item
- Some processes may occur concurrently while others must wait for some other process to complete.

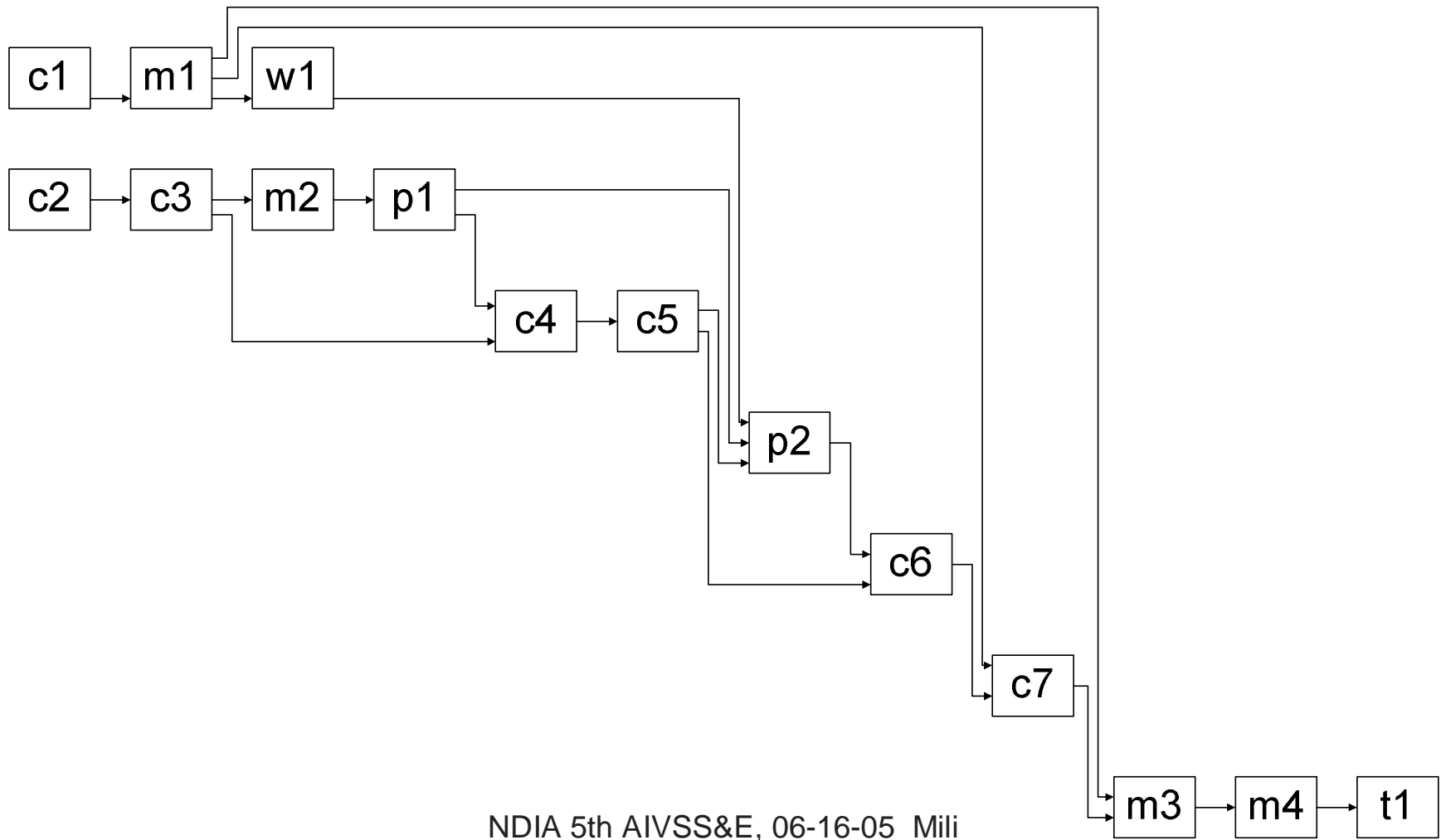


Example 2



```
(procedure
  (index (slow-move-click ?target))
  (step c1 (initiate-move-cursor ?target))
  (step m1 (move-cursor ?target)
           (waitfor ?c1))
  (step c2 (attend-target ?target))
  (step c3 (initiate-eye-movement ?target)
           (waitfor ?c2))
  (step m2 (eye-movement ?target)
           (waitfor ?c3))
  (step p1 (perceive-target-complex ?target)
           (waitfor ?m2))
  (step c4 (verify-target-position ?target)
           (waitfor ?c3 ?p1))
  (step c5 (attend-cursor-at-target ?target)
           (waitfor ?c4))
  (step w1 (WORLD new-cursor-location ?target)
           (waitfor ?m1))
  (step p2 (perceive-cursor-at-target ?target)
           (waitfor ?p1 ?c5 ?w1))
  (step c6 (verify-cursor-at-target ?target)
           (waitfor ?c5 ?p2))
  (step c7 (initiate-click ?target)
           (waitfor ?c6 ?m1))
  (step m3 (mouse-down ?target)
           (waitfor ?m1 ?c7))
  (step m4 (mouse-up ?target)
           (waitfor ?m3))
  (step t1 (terminate)
           (waitfor ?m4)))
```

Figure 2. PDL code for the CPM-GOMS template shown in Figure 1.



NDIA 5th AIVSS&E, 06-16-05 Mili



Motivation: Separation of Concerns



- ❑ Description of CE-6 focuses exclusively on the task at hand.
- ❑ Description of CE-6 refers to two separate roles: Radar controller and Radar Associate Controller.
- ❑ Detailed performance model makes explicit reference to agents performing the task, and to the interface being used.



Motivation: Reusability of Modeling Information



□ What to reuse?

- Definition of tasks
- Decomposition of tasks
- Knowledge/skills requirements of tasks
- Information about agents
- Information about interfaces
- Information about assignment of agents to tasks
- Information about fit between interfaces and tasks
- Etc.



Reusability in traditional performance Modeling environments



□ Reuse of

- Equipment
- Agents (training, expertise, etc.)
- Micro-models

□ Reuse of performance data about humans

- Single finger keying rate
- Cursor movement with mouse
- Rotary dial
- Hand movement (Fitt's law)
- Walking rate
- Simple reaction time, On or Off responses



Reusability lacking in Traditional Performance Modeling Environments



- ❑ “Macro” domain models.
- ❑ Domain models that are independent of
 - Agent assignment.
 - Equipment characteristics.



1.2 Requirements



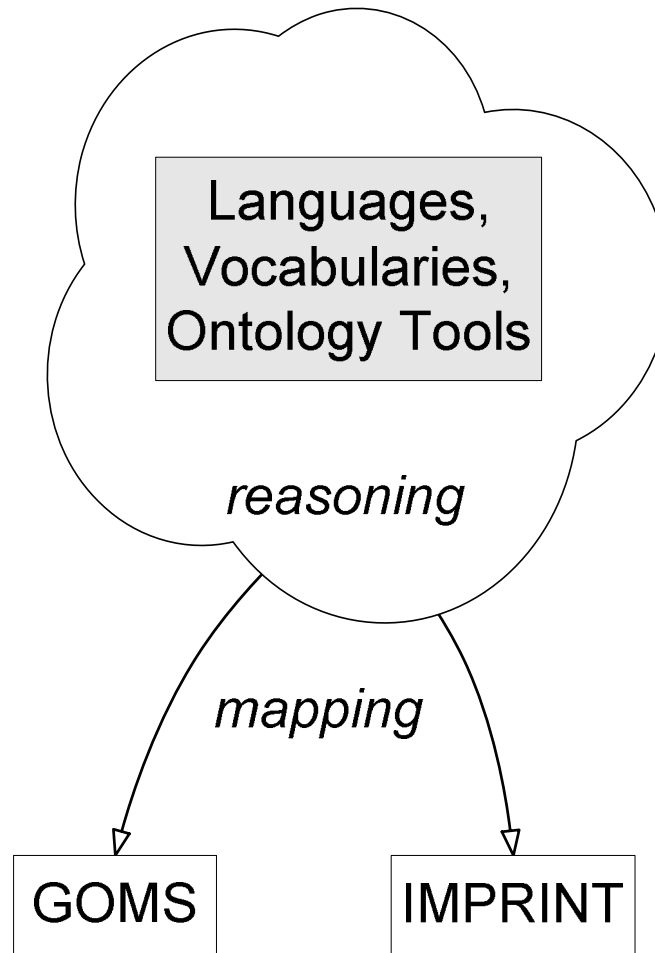
- ❑ Develop a methodology of system design support with the following features:
 - Support for the development and maintenance of a library of domain models.
 - Domain models must be sharable and reusable.
 - Domain models must support high level reasoning about missions, systems, and agents
 - Domain models must be good starting points for detailed performance models.



Components of methodology



1. Language, representation for capturing the domain model.
2. Tools for constructing and maintaining domain model.
3. Reasoning and analysis tools for domain models.
4. Mappings from domain model components to performance models





Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models.
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Reusing Vocabularies and Ontologies
 3. Tools for constructing and maintaining domain model.
4. Summary Conclusion



2.1 Ontology of Tasks



What information do we need about tasks?

- Data transformation performed by the task [example](#)
 - Input
 - Output
- State transformation performed by a task [example](#)
 - Precondition
 - Effect
- Resources required to perform the task [example](#)
 - Nature, quantity
- Structural information
 - What are the components of a task
 - How the components relate to each other



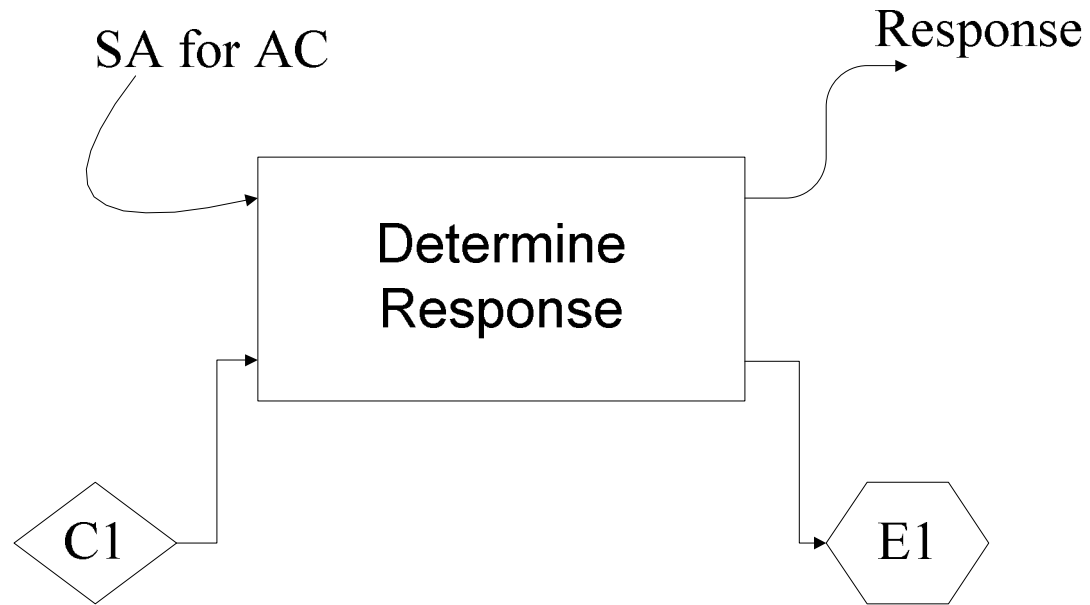


Input/Output Information



□ Example

□ [back](#)



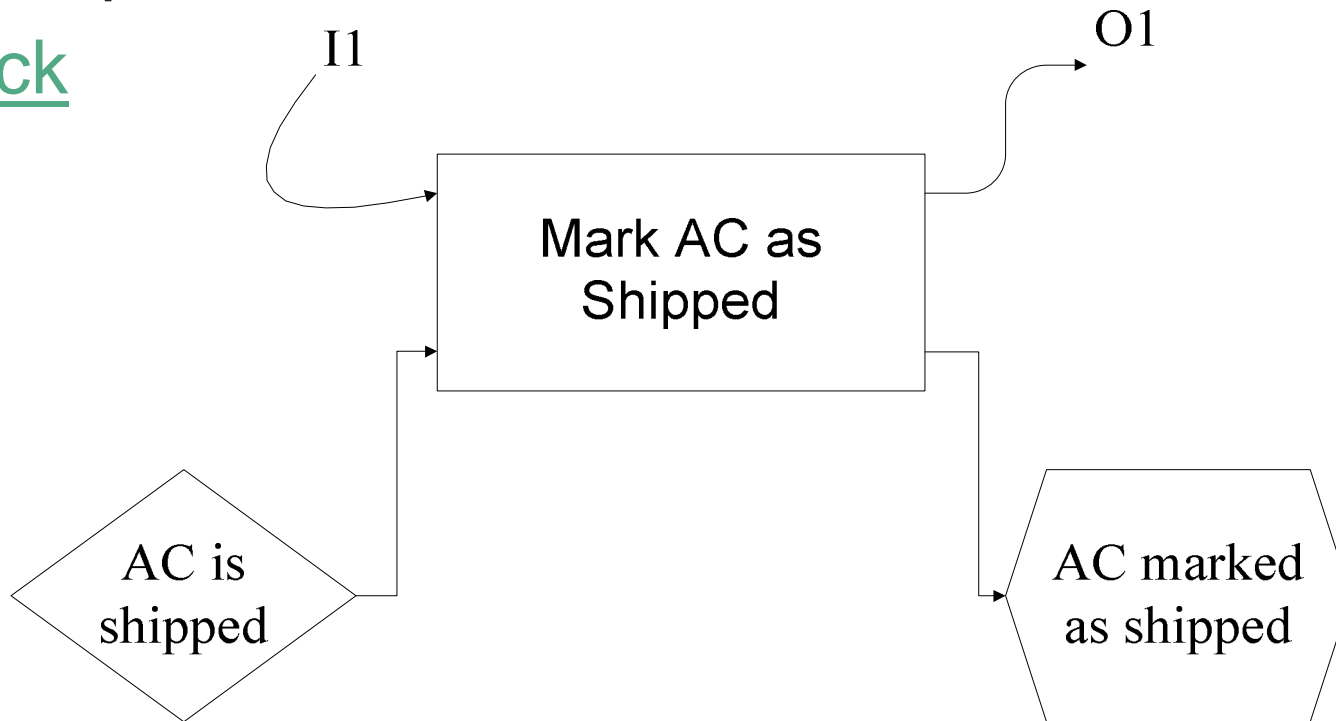


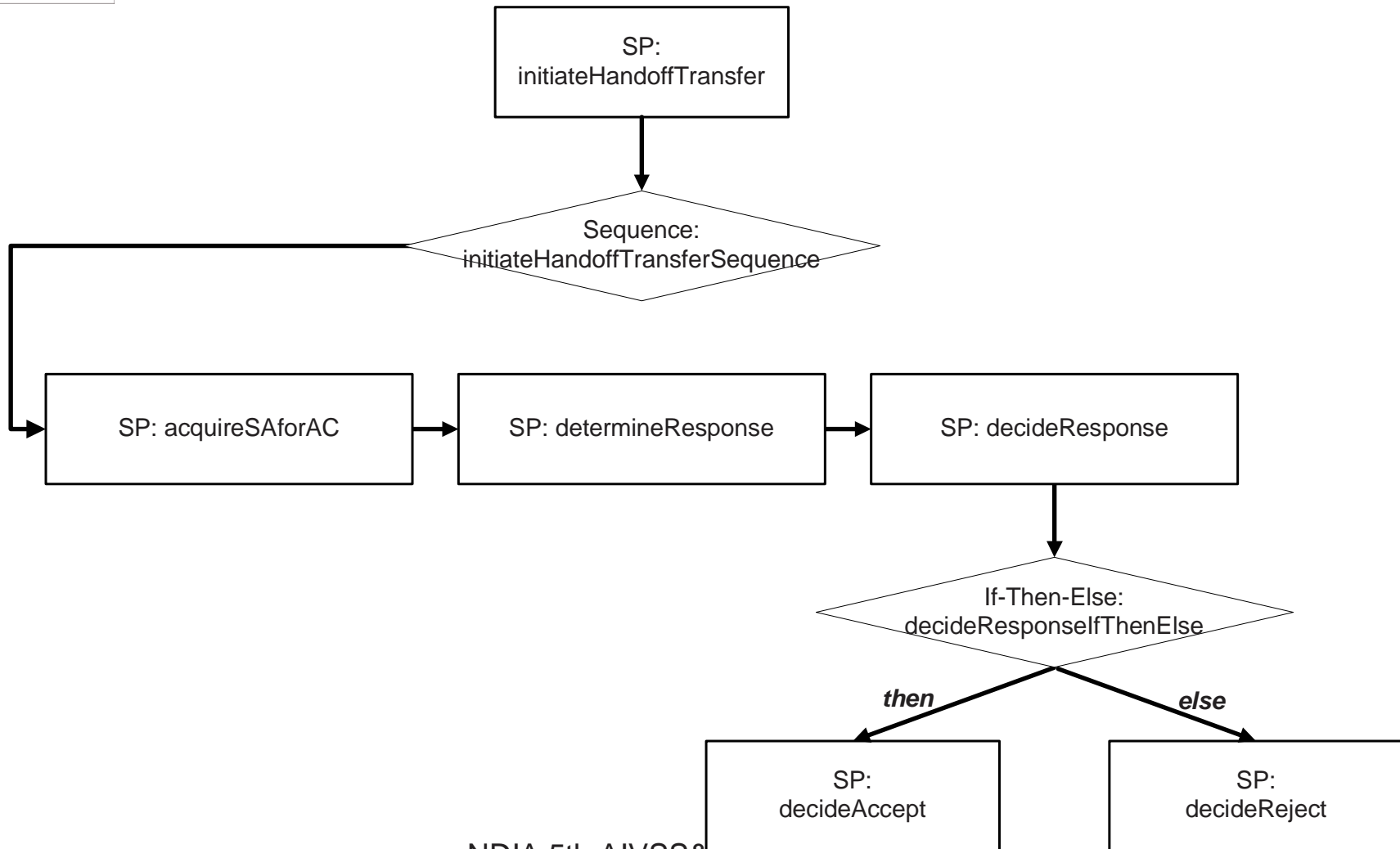
Precondition/Effect



Example

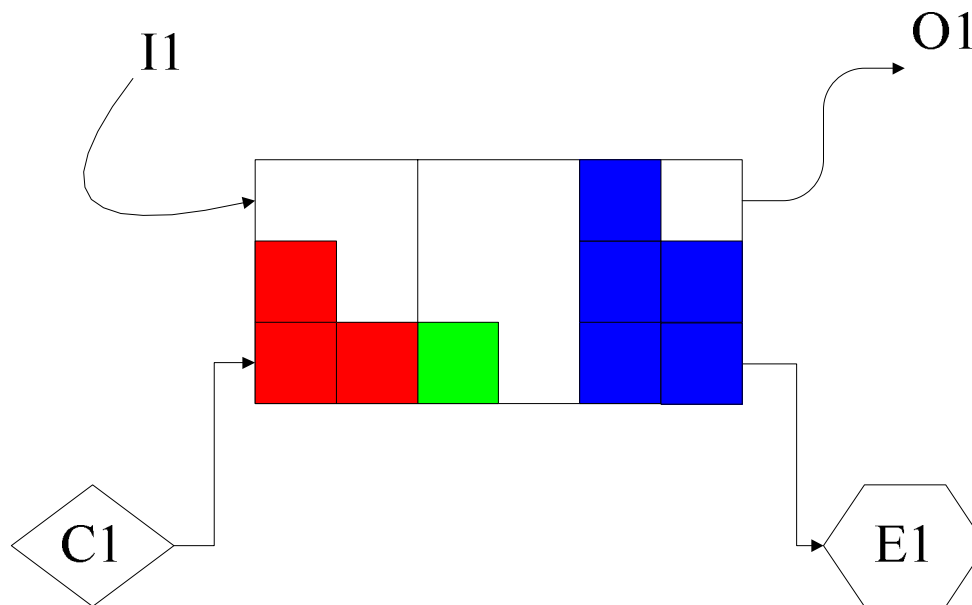
back







Atomic Task





Atomic Process



- The most basic process
- Cannot be decomposed



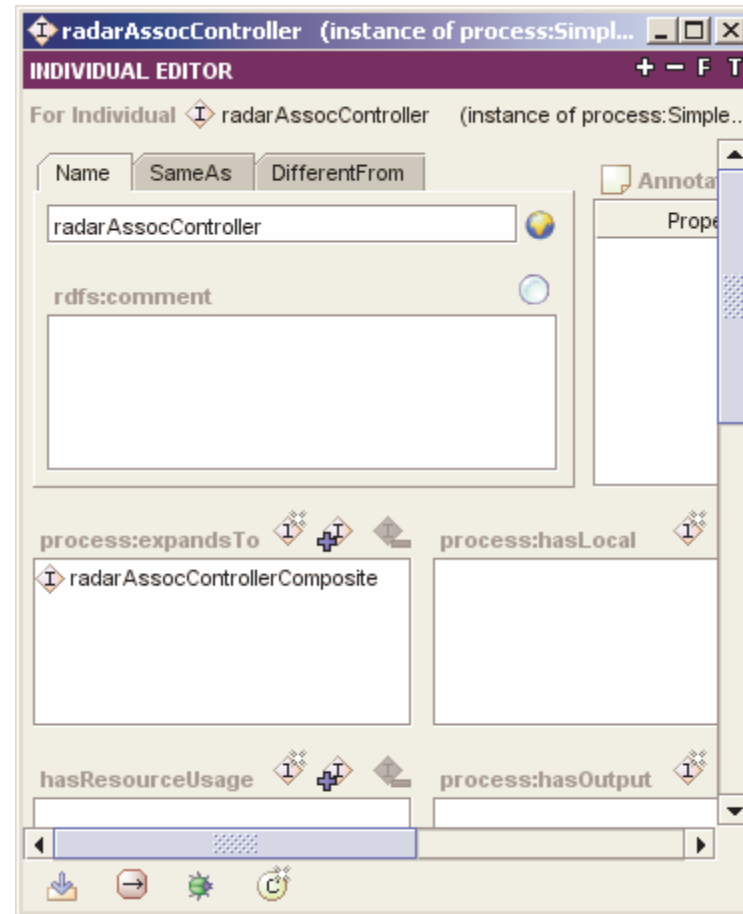
Simple Process



- Can be realized by an atomic process
- Can stand alone, waiting for refinement
- Can expand to a composite process



SimpleProcess expandsTo CompositeProcess





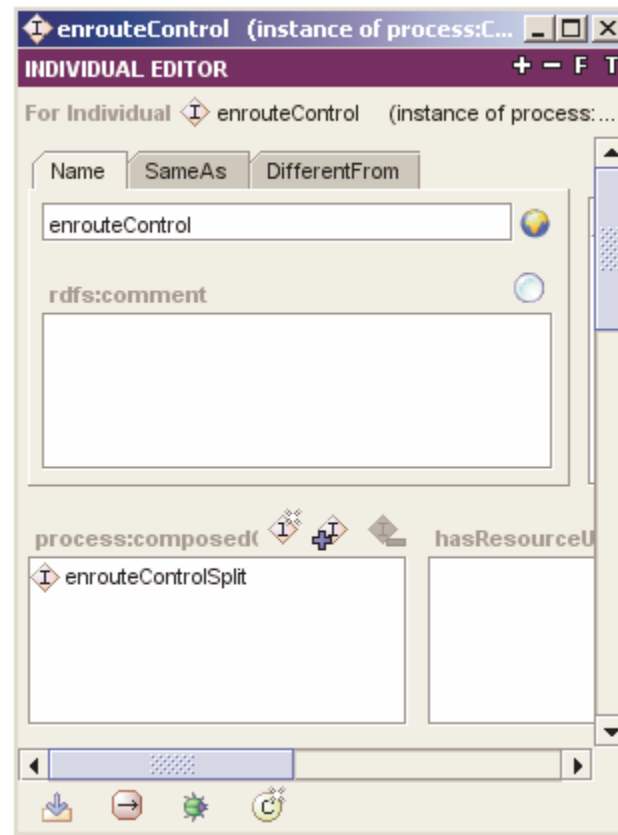
Composite Process



- Collapses to a simple process
- Is a refined view of that simple process
- Is composed of a control construct (split, sequence, etc.)



CompositeProcess composedOf ControlConstruct





Control Construct



- ❑ Governs the execution order of the processes within a composite process
- ❑ Determines the logical relationship and dependencies between components.

Control Construct	Relationship
Any-Order	Components can be executed in any order
If-Then-Else	Choice of one process or another
Sequence	Time-ordered list of processes
Split	Concurrent execution
Split-join	Concurrent processes with synchronization



Resources



- All processes use resources that have a quantity and a type (cognitive, motor, or perception).
- Simple processes and composite processes also have computed resource usages.



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models.
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Reusing Vocabularies and Ontologies
 3. Tools for constructing and maintaining domain model.
4. Summary Conclusion



Semantics of control constructs



For each control construct, we specify

Consistency Condition

Axioms for deriving

➤ Input

➤ Output

➤ Preconditions

➤ Effect

➤ Resource usage

Of composite tasks



Semantics of Sequence



T is a sequence with components $T_1, T_2 \dots, T_n$

Consistency Rule

The effect of task T_i must not conflict with the precondition of task T_{i+1} .

Precondition and Effect

The precondition of T is (at least) the precondition of T_1 .

The effect of T is the (at least) the effect of T_n .

Resource Usage

The resource usage of T is the sum of resource usage by $T_1, T_2 \dots, T_n$



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models.
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Reusing Vocabularies and Ontologies
 3. Tools for constructing and maintaining domain model.
4. Summary Conclusion



Enforcing Consistency



- ❑ When are we checking consistency?
 - Whenever a simple task is expanded
 - Whenever a component is modified
- ❑ What to check?
 - Preconditions and effects (of components of the same composite)
 - Preconditions and effects (of composite task relative to those of its components)
 - Inputs and outputs (of a composite task relative to those of its components)
 - Resource usage (of a composite task relative to those of its components)

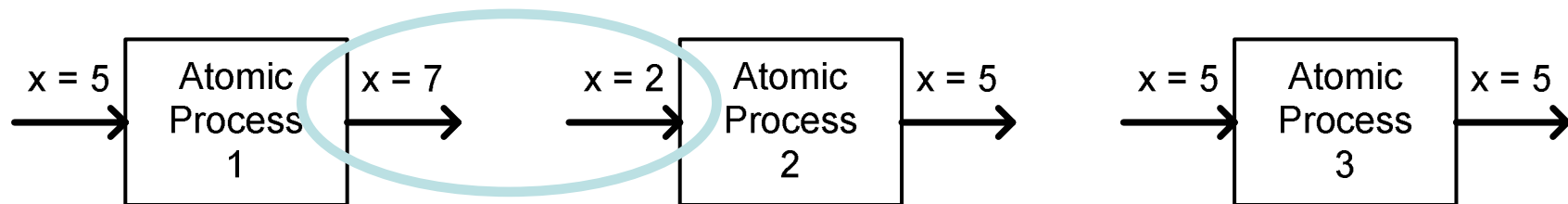


Preconditions and effects



Decomposition	Consistency Formula
Sequence	The precondition of each component is satisfied by the effect of the subsequence preceding it.
Any order	Components are independent and mutually compatible.
Split	The preconditions of the components do not conflict
If-Then-Else	

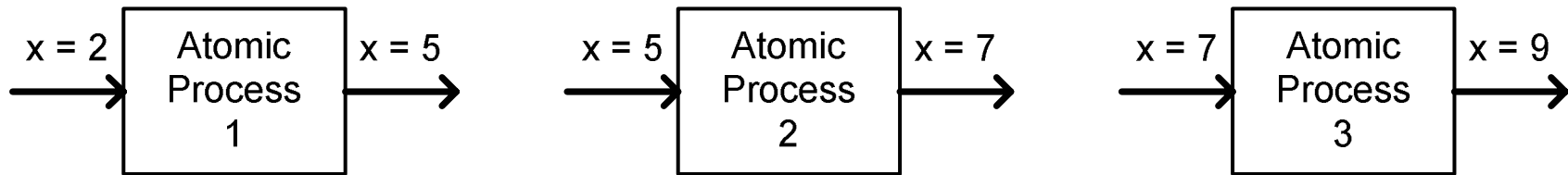
Preconditions and Effects, cont.



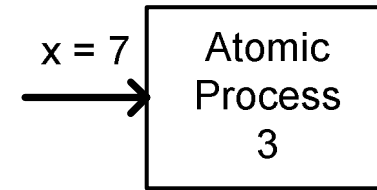
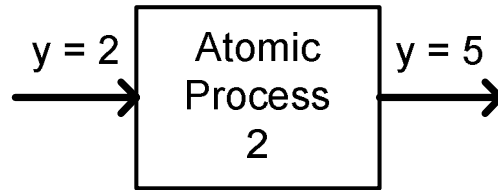
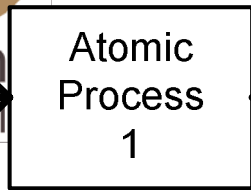
- This sequence has a *conflict* because Atomic Process 1 has an effect that $x=7$ and is followed by Atomic Process 2 that has a precondition that $x=2$.



Preconditions and Effects, cont.

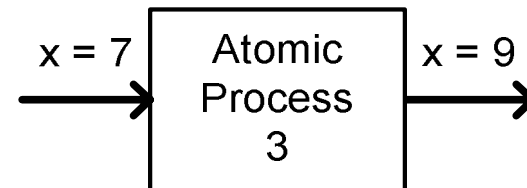
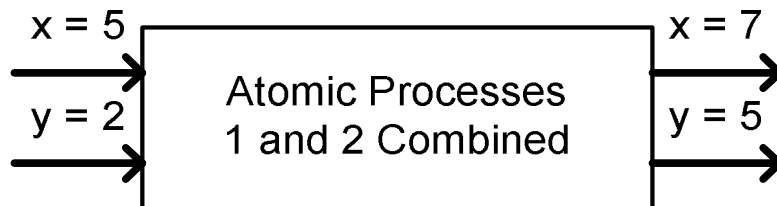


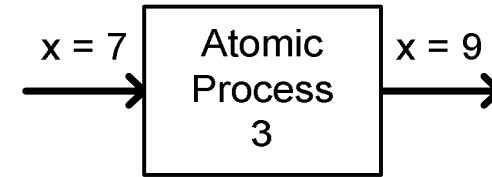
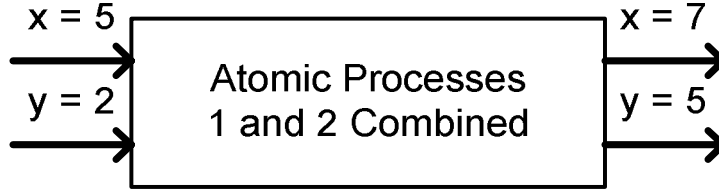
- This sequence *does not have a conflict* because all preconditions are satisfied.



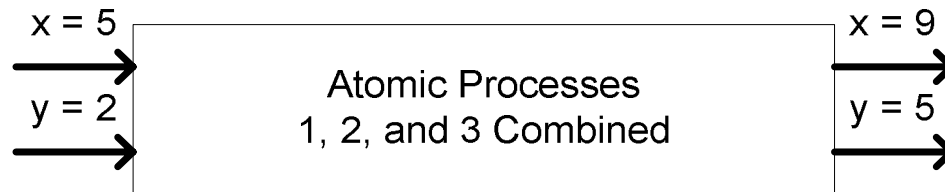
□ Given the above sequence

□ Can combine the first two processes

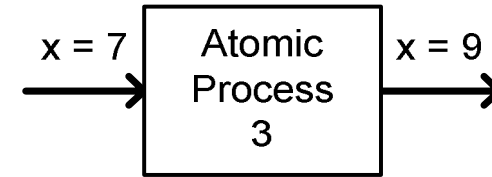
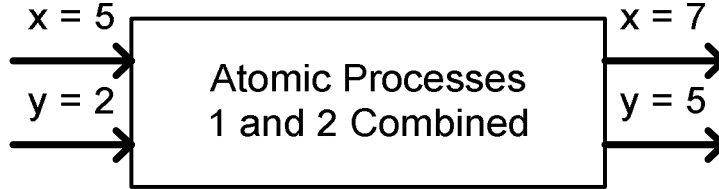




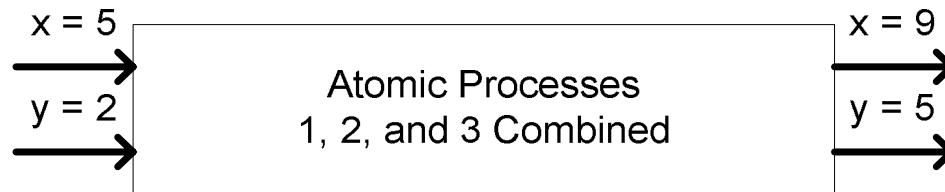
- The remaining process can be combined with the first two.



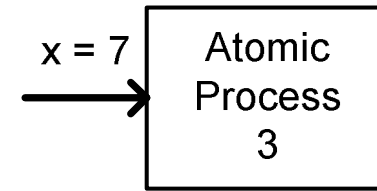
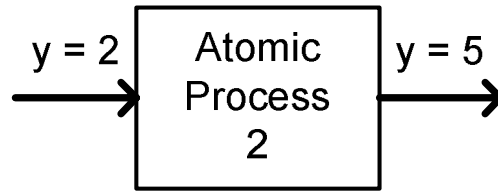
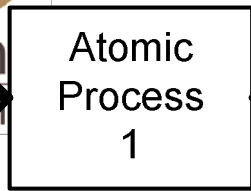
- This represents the overall precondition and effect for the entire composite process



- The remaining process can be combined with the first two.

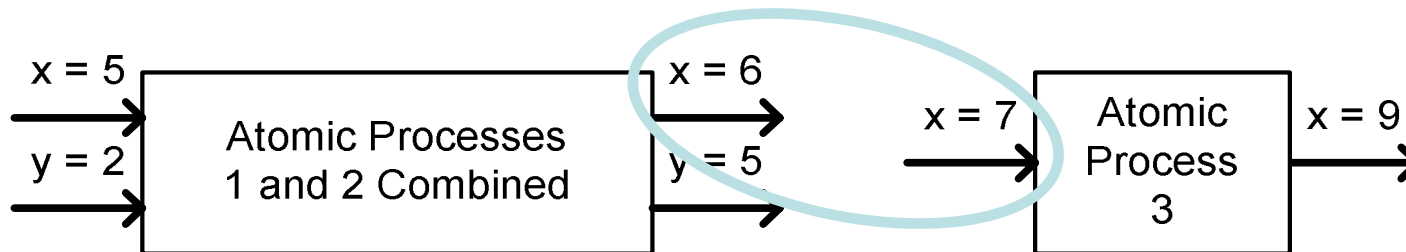


- This represents the overall precondition and effect for the entire composite process



□ As with the previous example, we can combine the first two processes.

□ Here, we have a conflict





Computing Values



- ❑ Focus on resource values.
- ❑ Resource values differ from other parameters (input, output, preconditions, effects) because they are tentative.
- ❑ There may be different estimates from different sources, e.g. direct estimate vs. computed from estimates of components.
- ❑ It may make sense to keep different values keeping track of their sources.
- ❑ Information about sources will be useful in deciding relative credibility.



Computing values, Example



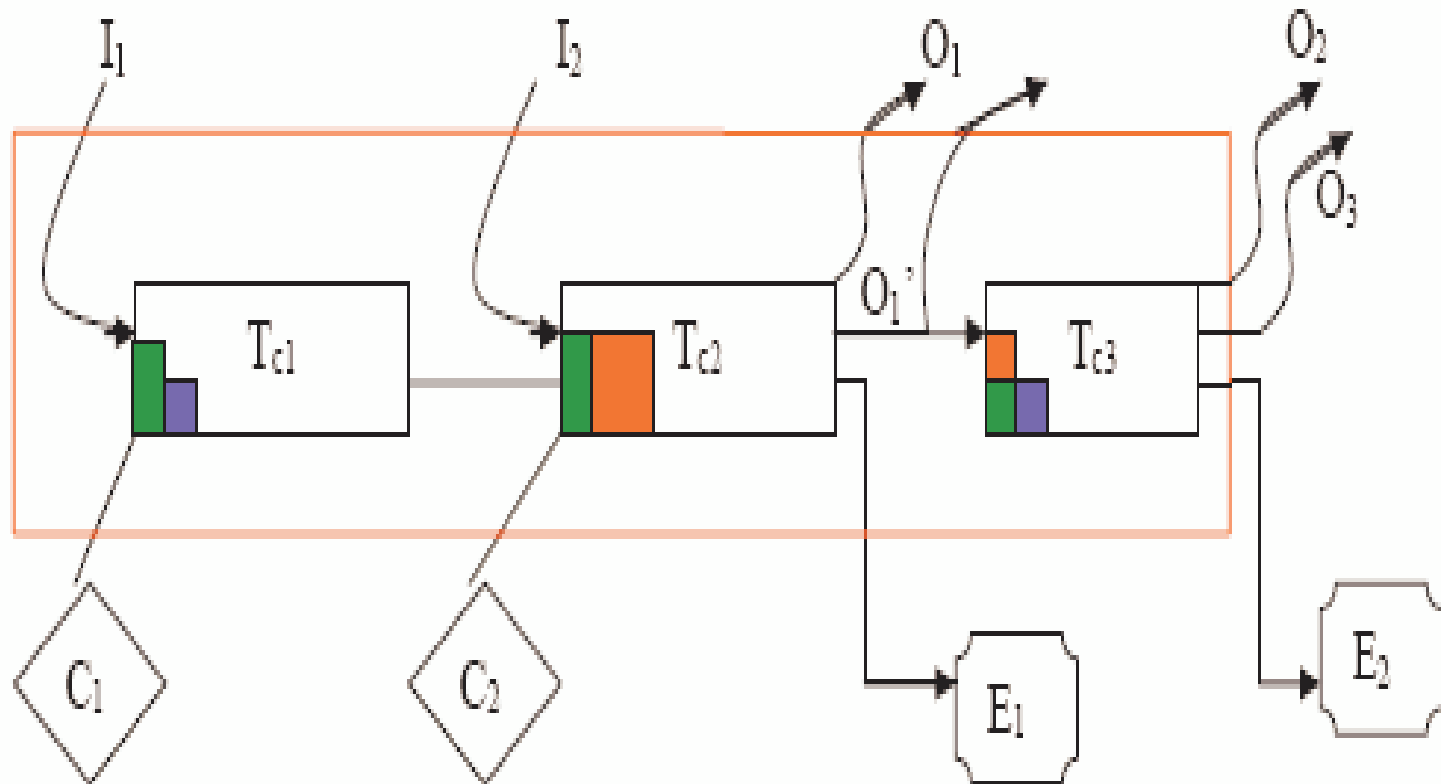
We consider the simple task T_s characterized as follows

- Inputs: I_1, I_2
- Outputs: O_1, O_2, O_3
- Preconditions: C_1, C_2
- Effects: E_1, E_2
- Resources: $\{ \langle C, 4 \rangle, \langle M, 2 \rangle, \langle P, 5 \rangle \}$



Task decomposed into a sequence of Tc1, Tc2, Tc3:

		T_{c1}	T_{c2}	T_{c3}
Input		I_1	I_2	O_1'
Output			O_1, O_1'	O_2, O_3
Preconditions		C_1	C_2	
Effects			E_1	E_2
Resources	C	2	2	1
	P	1	0	1
	M	0	4	1





Computing Resource values cont'



- All tasks have resource usage information.
- All composite tasks (simple, expanded) have also computed resource usage information.
- Computed values and values entered directly may or may not match.

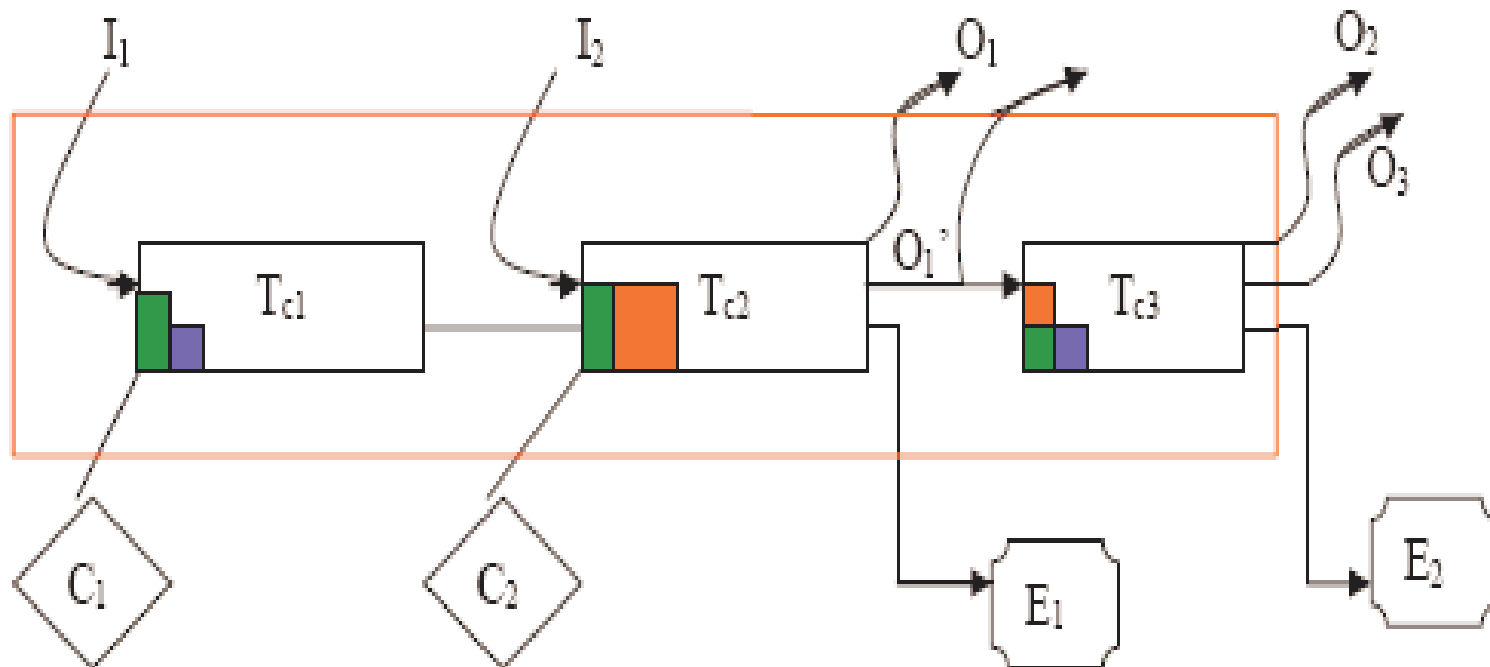


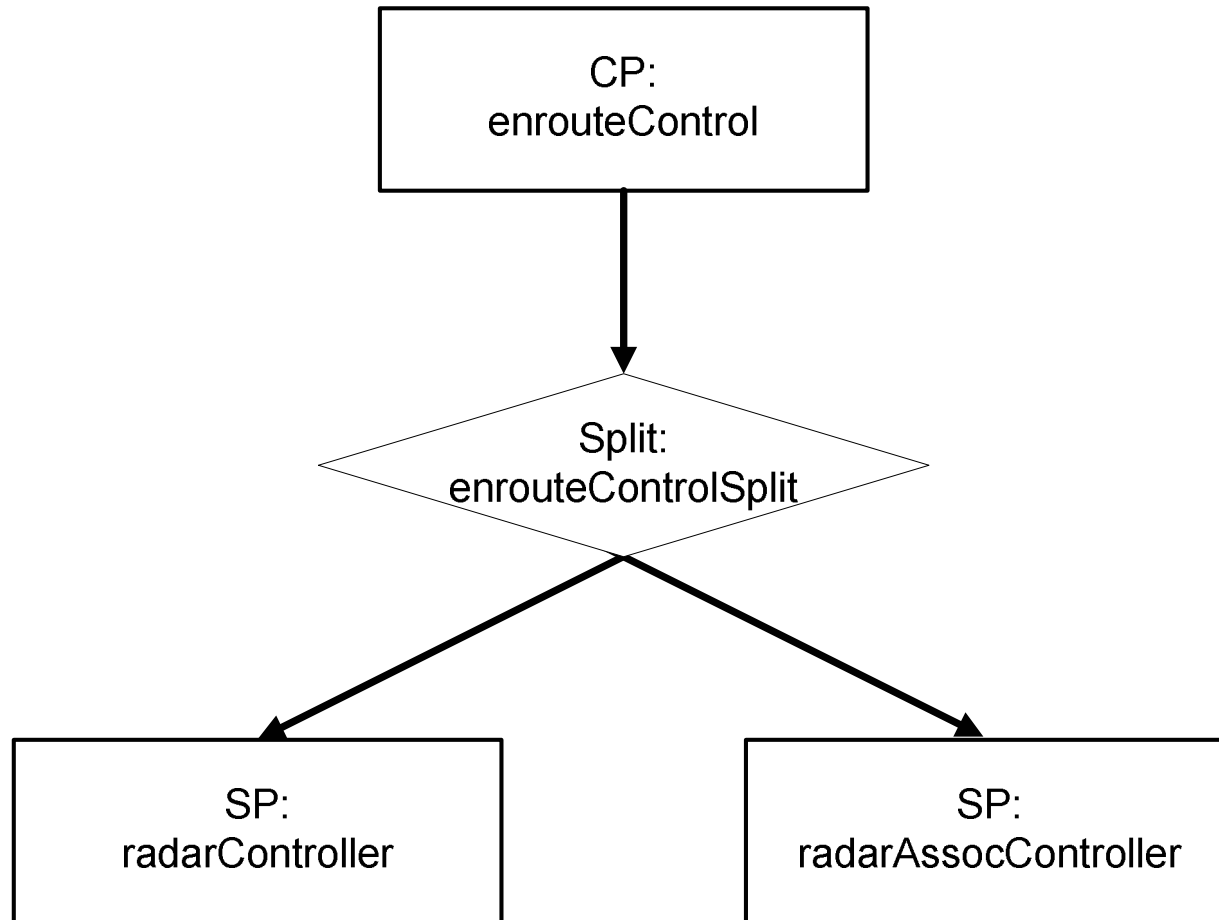
Resource computation

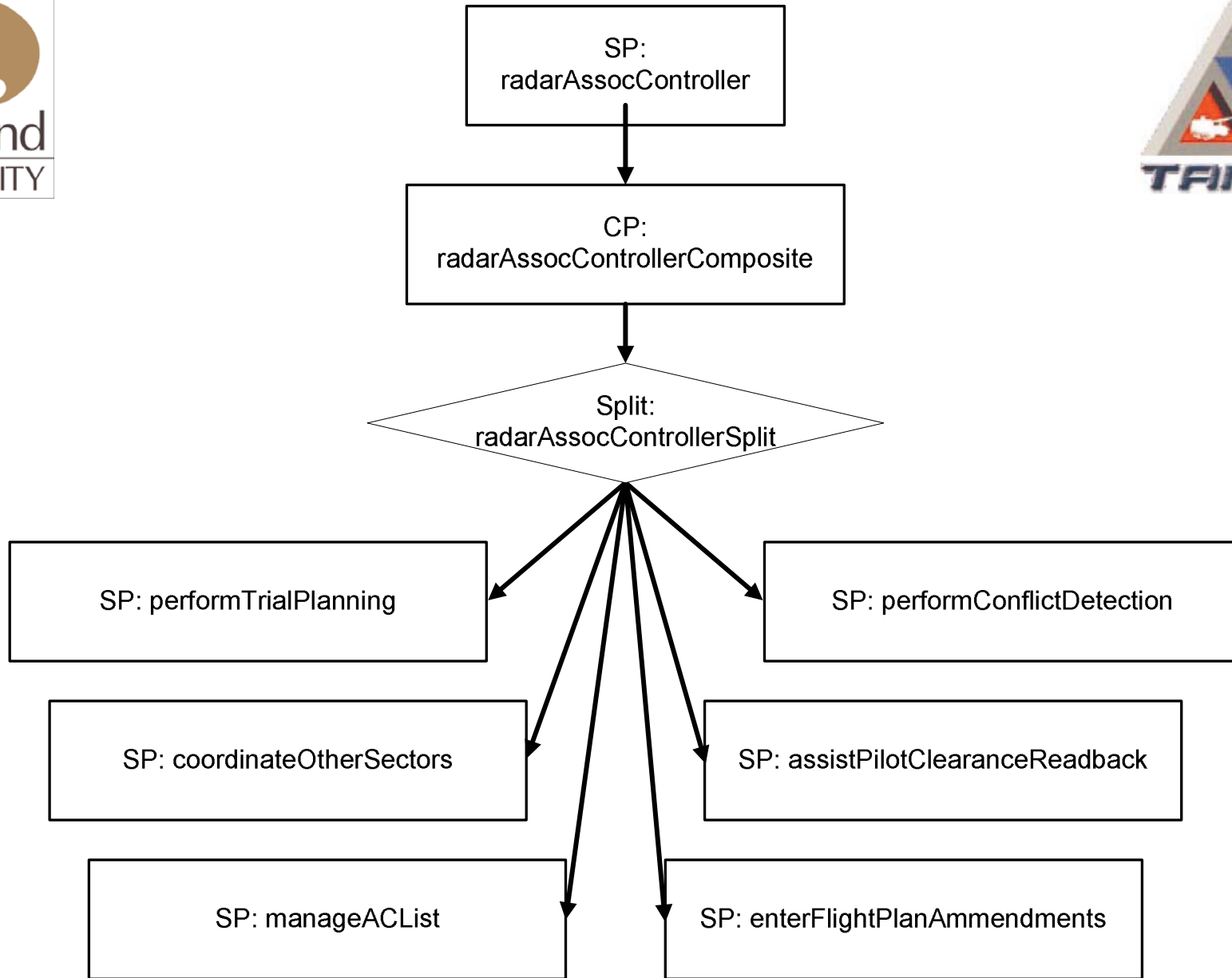


decomposition	Computed Resource
Sequence	For each resource type, $R = \text{Sum of resources of components}$
Any Order	For each resource type, $R = \text{Sum of resources of components}$
Split	For each resource type, $R = \text{Sum of resources of components}$
If-Then-Else	For each resource type, $R = \text{Max of resources of components}$

5.2 Task decomposition and Consistency checking

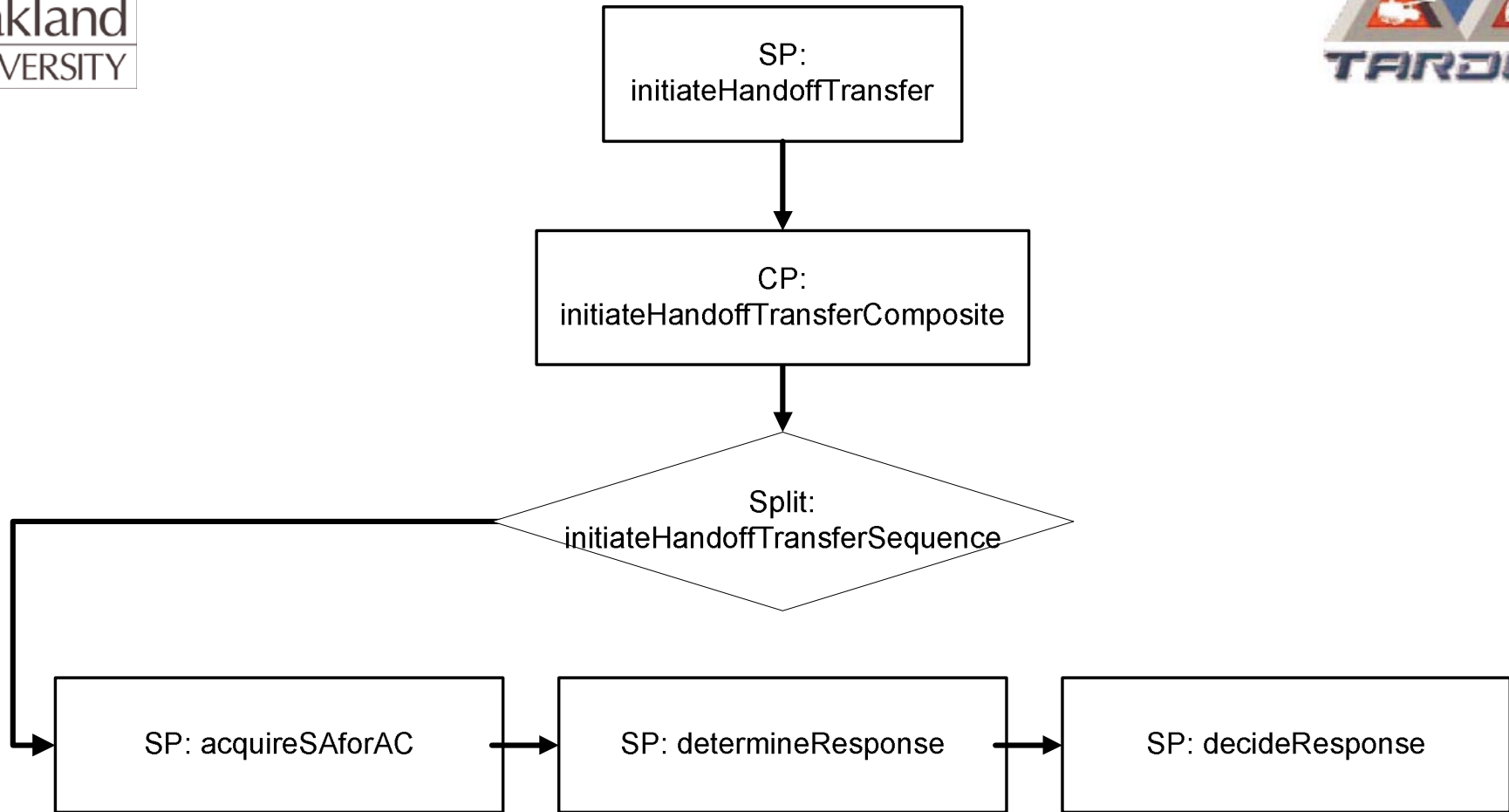






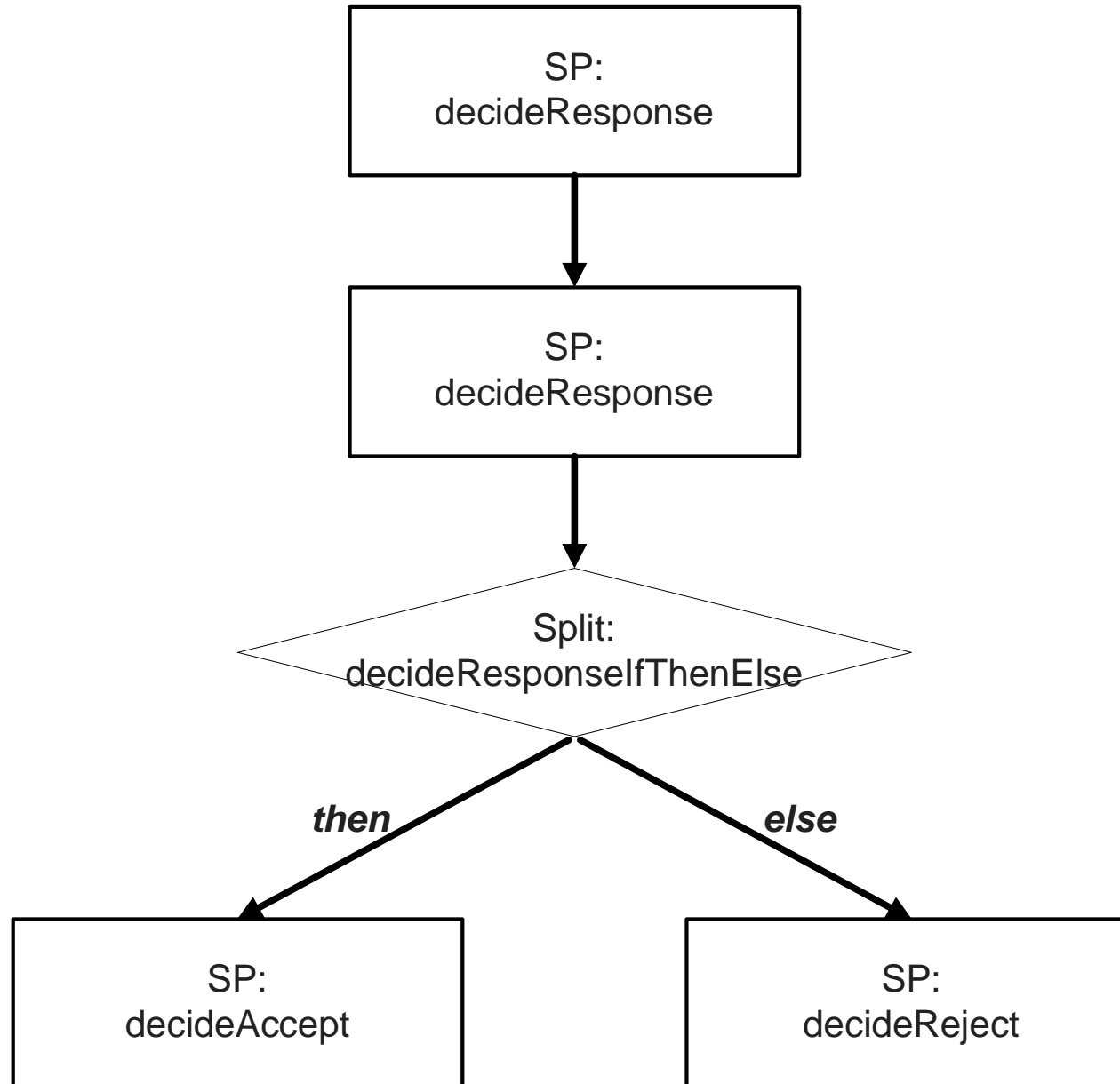


Process	Cognitive Resource	Motor Resource	Perception Resource
performTrialPlanning	30	20	15
coordinateOtherSectors	40	10	20
manageACLlist	10	25	5
enterFlightPlan Ammendments	5	25	5
assistPilotClearance Readback	15	5	10
performConflict Detection	20	5	10
radarAssocController Composite	120	90	65





Process	Cognitive Resource	Motor Resource	Perception Resource
acquireSAforAC	10	10	15
determineResponse	20	5	10
decideResponse	30	5	20
initiateHandoff TransferComposite	60	20	45





Process	Cognitive Resource	Motor Resource	Perception Resource
decideAccept	30	5	15
decideReject	20	5	20
decideResponse	30	5	20



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Reusing Vocabularies and Ontologies
 3. Tools for constructing and maintaining domain model.
4. Summary Conclusion



Reasoning about Domain models



Consider task T refined with the end result that T uses (100, 60, 85) resources.

This information is similar to the information that a software project takes 50 person-month.

It is a reflection of the complexity of the task.

It says nothing about the level of parallelism possible within the task

The structure of the task sheds more light on

- how quickly we can complete such a task
- How many agents should be assigned



How long does it take to execute
T structured as follows?



T_{11}	T_2	T_{311}	T_{3121}
T_{12}			T_{3122}
T_{13}			T_{3123}
		T_{312}	
		T_{312}	

- ❑ Simple scenario 1: Assign only one agent. All tasks are executed sequentially.



How many agents are needed?



Agent 1				T3121
Agent 2				T3122
Agent 3	T11		T311	T3123
Agent 4	T12		T312	
Agent 5	T13	T2	T313	

Simple scenario 2: Assign a number of agents equal to the breadth of the decomposition tree



Task T= sequence (T1, T2)



T1

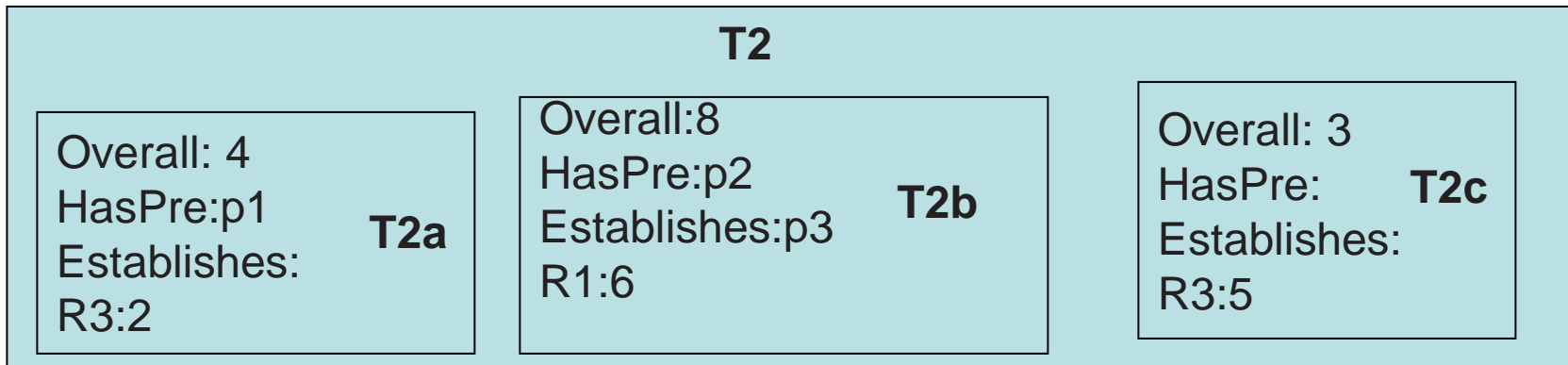
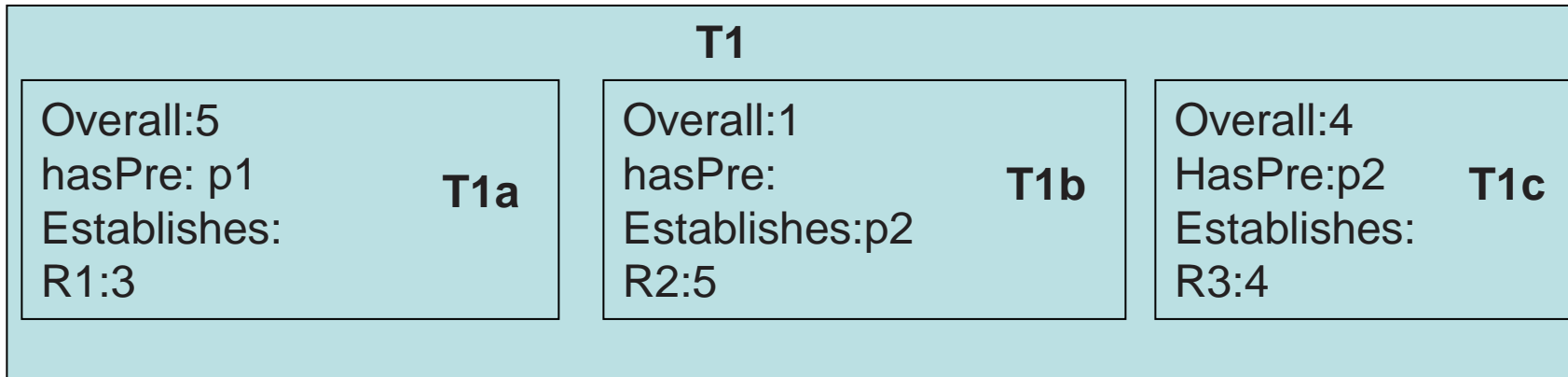
OverallTime:10
Has precondition:P1
Has effect: P2
ResourceUsage:
 R1: 3
 R2: 5
 R3: 4

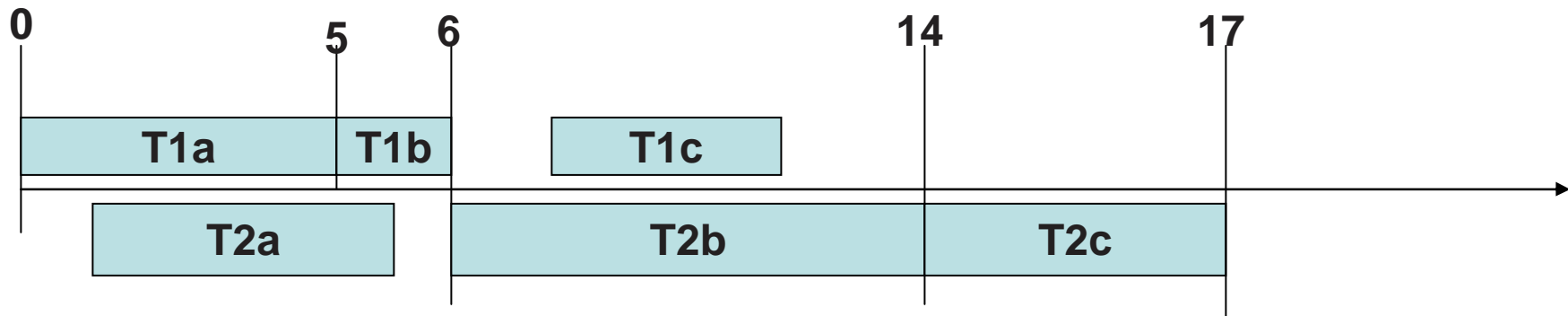
T2

OverallTime:15
Has precondition: P2
Has Effect: P3
ResourceUsage:
 R1:4
 R2:3
 R3:5

T

OverallTime:25
HasPrecondition:P1
Has Effect: P3
ResourceUsage:
 R1:7
 R2:3
 R3:9





17 vs 25

Information used: Preconditions, post-conditions, resource usage, and constraints on resource usage.



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Tools for constructing and maintaining domain model.
 3. Language, representation for capturing the domain model.
4. Summary Conclusion



Domain Models to Performance Models



❑ Roles vs. agents

- Tasks are assigned to roles (e.g. driver, controller) in domain models
- Actions are assigned to specific agents in performance models

❑ Bridging the gap between OWL-S models into IMPRINT, CPM GOMS

- Add agent definitions and interface specification
- Each model task can be refined into model
- Translating resource values



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models
 5. Mappings from domain model components to performance models
3. **Technological view (white box)**
 1. Language, representation for capturing the domain model.
 2. Tools for constructing and maintaining domain model.
 3. Language, representation for capturing the domain model.
4. Summary Conclusion



Technological view



- ❑ What is inside the black box?
- ❑ What language is used?
 - XML, RDF, RDFS, OWL, OWL-S, DL, ... why so many?
 - What is imported? What is created from scratch? What is reused?
 - What tools are used to edit, maintain, update the domain model?



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models
 5. Mappings from domain model components to performance models
3. **Technological view (white box)**
 1. **Language, representation for capturing the domain model.**
 2. Tools for constructing and maintaining domain model.
 3. Language, representation for capturing the domain model.
4. Summary Conclusion



1. Language for capturing Domain model



Uniform alphabet → XML

Uniform syntax → RDF



- ❑ Domain Models must be sharable, reusable, thus,
 - Use language universally known and understood (XML);
 - Describe domain models so that they can be easily located (RDF) and uses well defined vocabularies (RDFS)
 - Reuse existing domain models (OWL-S);
- ❑ Domain models must support high level analysis and reasoning,
 - Use a language with well defined semantics (DL)



We use



- XML encoding
- RDF syntax
- RDFS-defined vocabulary
- OWL-S upper ontology as a basis
- DL as a semantics
- Protégé as the editing tool.



XML/RDF



- Becoming the standard notation for the semantic web.
- Benefits from an increasing number of tools for displaying and editing it.
- Can be directly manipulated by applications.



XML, an SGML markup language



Standard Generalized Markup Language (SGML)

Goal:

- international standard for the definition of
- device-independent,
- system-independent
- methods of representing documents in electronic form.

Will introduce XML by relating it to SGML, HTML



SGML characteristics



□ Descriptive vs. Procedural

This is a paragraph:

There are three characteristics of SGML which distinguish it from other markup languages.

Indent here by .5'

There are three characteristics of SGML which distinguish it from other markup languages.

End the line and skip a line.



SGML characteristics cont'



□ Document type.

- Documents have document types (DTD).
- DTD defines components, structure, etc.
 - E.g. slide has a title and a body. Body has a bulleted list.
- Parsers use the DTD to interpret/process documents.
- Documents of same type can be processed using the same parser (e.g. browser).



SGML characteristics cont'



Data Independence

- Documents are portable to different hardware and software systems.
- Different character encodings are addressed by descriptive mappings of non-portable characters.



SGML Languages: HTML



- ❑ HTML: SGML Language with predefined DTD.

- ❑ An HTML document

 - `<html>`

 - `</html>`

- ❑ An HTML Document has a head and a body

 - `<html>`
 - `<head>`
 - `<title>`
 - `</title>`
 - `</head>`
 - `<body>`
 - `</body>`
 - `</html>`



Example 3 HTML



Show file source and file displayed in a browser.



HTML cont'



- ❑ HTML: very simple and powerful idea. Simple syntax, enabled the web as we know it.
- ❑ HTML philosophy:
 - Specify what is, (title, paragraph, table, rows, etc.)
 - Let browsers decide how to display (selecting font, size, placement, ...)
 - Documents can be characterized by their contents:
 - o Find documents with “theory of computation” in the title.
 - o Find documents with paragraphs starting with “Once upon a time.” etc.



HTML Characteristics



- ❑ Pre-defined set of tags.
- ❑ This allowed browsers (parsers) to be developed for html for all platforms.
- ❑ HTML as an SGML language, was meant to annotate *contents* (body, paragraph, quote, etc.) not *format* (line breaks, fonts, etc.).
- ❑ Temptation for writers to control format violated the general philosophy. HTML tags include data formatting tags.



HTML strengths and limitations



Predefined set of tags

- Can control what is and what is not a valid tag
- Can develop parsers to process any html document

Predefined set of tags

- Structure and contents that can be described with the tags is limited.
- Most of the document is inaccessible to automatic parsers.
- Mostly adapted to documents destined for the human eye.



Other issues with HTML



- The predominance of HTML documents, which mix content with presentation
- The difficulty of maintaining Web sites to reflect inevitable real-world changes
- The difficulty of seamlessly presenting dynamic content
- The seeming futility of finding precisely what one wants using a Web-crawler search engine.



Semantic-Web



- Accessing web by contents rather than just by keywords.
- Development of a new generation of Web markup languages.
- Use of these languages to describe data available on the web.
- Use of these languages to capture semantics.
- Development of tools for searching, interpreting, presenting this data.



We use



- XML encoding
- RDF syntax
- RDFS-defined vocabulary
- OWL-S upper ontology as a basis
- DL as a semantics
- Protégé as the editing tool.



Extensible Markup Language



[7]

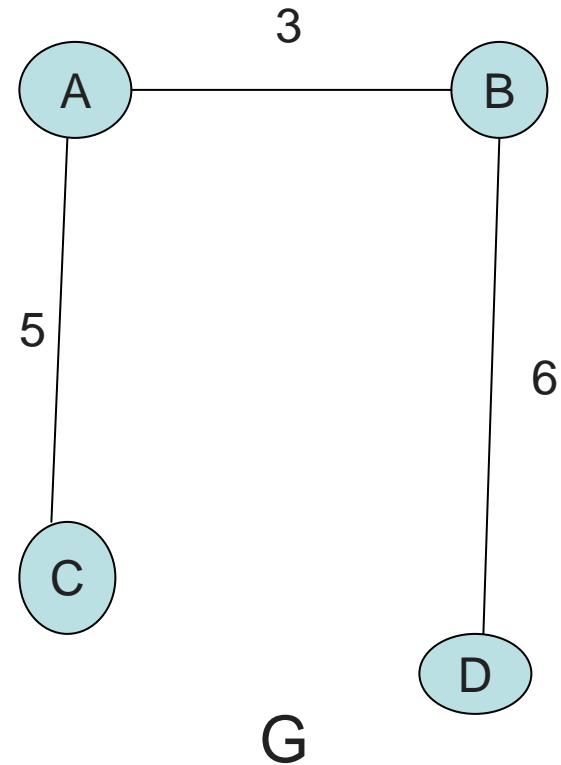
- XML stands for **EX**tensible **M**arkup **L**anguage
- XML is a **markup language** much like HTML
- XML was designed to **describe data**
- XML tags are not predefined. You must **define your own tags**
- XML uses a **Document Type Definition (DTD)** or an **XML Schema** to describe the data
- XML with a DTD or XML Schema is designed to be **self-descriptive**
- XML is a W3C Recommendation



Example 4



```
<graph>
<name> G </name>
  <edge>
    <vertex> A </vertex>
    <vertex> B </vertex>
    <distance> 3 </distance>
  </edge>
  <edge>
    <vertex> A </vertex>
    <vertex> C </vertex>
    <distance> 5 </distance>
  </edge>
  <edge>
    <vertex> B </vertex>
    <vertex> D </vertex>
    <distance> 6 </distance>
  </edge>
</graph>
```





Example 2: html vs. xml



```
<ol>
<li>John Smith <it>How
to compose html</it>
MIT Press, Boston
MA, 2000. </li>
<li> Jane Smith <it>My
XML book</it>
Addison Wesley 4th
Edition, 2003.
</li>
</ol>
```

```
<booklist>
  <book>
    <author>John Smith</author>
    <title>How to compose html</title>
    <publisher>MIT Press</publisher>
    <place> Boston MA </place>
    <year>2000 </year>
  </book>
  <book>
    <author>Jane Smith <author>
    <title>My XML book</title>
    <publisher>Addison
    Wesley</publisher>
    <edition> 4th Edition</edition>
    <year> 2003 </year>
  </book>
</booklist>
```



Characteristics of XML



- Data can be described using domain specific tags.
- Data is self-describing, even without a DTD.
- Allows a more detailed description of the data.
- Style sheets can be defined to specify how to display the data.
- Query languages have been developed to query XML data.



Examples X-query



Get title, author and year for any book whose author's last name is smith or publisher is MIT Press

```
for $x in doc("books.xml") /booklist/book
where (contains($x/author/name/Last,"Smith") or
contains($x/publisher,"MIT Press"))
return ($x/title, $x/author,$x/year)
```



Strengths and weaknesses of XML



❑ Flexible markup language

- Can be adapted to the level of details desired. A book can be annotated at the word level or at the chapter level.
- Tag names drawn from the domain's vocabulary.

❑ Flexible markup language

- Likelihood that two document authors use the same tags is low. Too much variability.
- Hierarchical (nesting) structure is often artificial. Different authors likely to use different structures.



Limitations of XML



- ❑ No deduction supported.
 - XML document has flights, I am looking for trips.
 - XML document says x parent of y and y parent of z. What is the relationship between x and z?
 - XML document tells about diseases; I am looking for something about illnesses.



Limitations of XML



- ❑ HTML documents have a header and a body.
- ❑ HTML headers contain meta-data.
- ❑ Meta-data is used to search, among other things.
- ❑ XML documents focus exclusively on the data.
- ❑ Need a mechanism to describe the data *document*.
 - Meta data includes: author, where located, when updated, etc.
 - Meta data includes: which relations are transitive, how relations relate to other relations



We use



- XML encoding
- RDF syntax**
- RDFS-defined vocabulary
- OWL-S upper ontology as a basis
- DL as a semantics
- Protégé as the editing tool.



Resource Description Language (RDF)



- ❑ Language for creating the phonebook of the web.
- ❑ The idea of cataloguing contents started with PICS [8].
- ❑ The Platform for Internet Content Selection (PICS) started in 1995 to communicate ratings of Web pages.
- ❑ These ratings cover any aspect deemed relevant, e.g.
 - whether content is peer reviewed research, authored by an accredited researcher, etc.
 - contains sex, nudity, violence, foul language, etc.



PICS to RDF



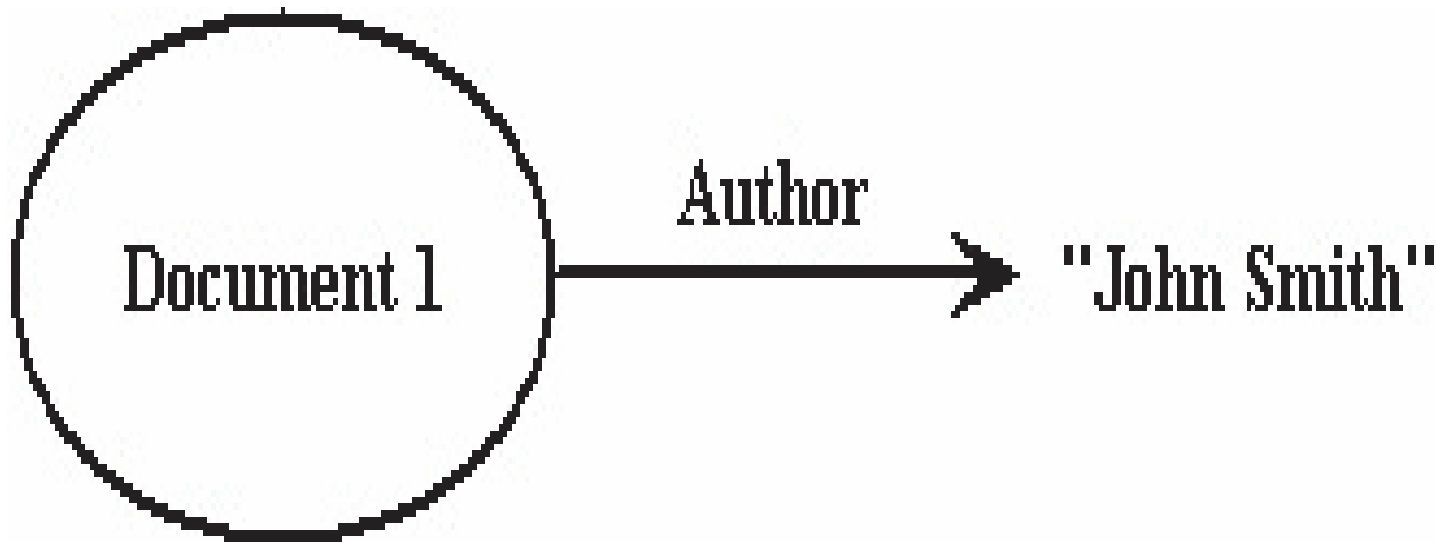
- ❑ PICS did not have a fixed set of criteria. Any “client” can define criteria they want to use.
- ❑ Specifications of PICS led to generalize the concept to “Resource Description Framework, RDF.

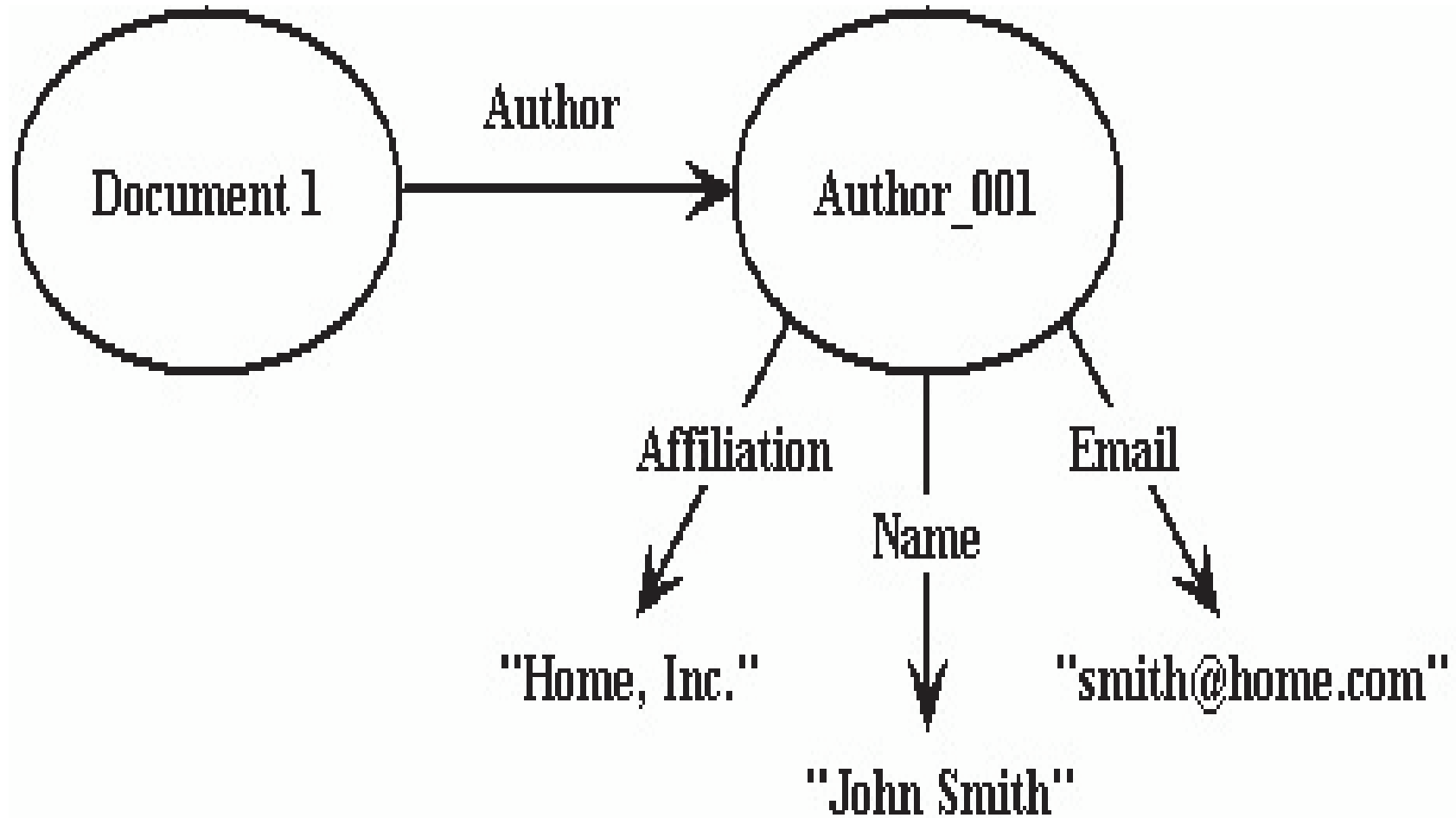


Example [9]



- John smith authored Document 1
- The author of document 1 is John Smith





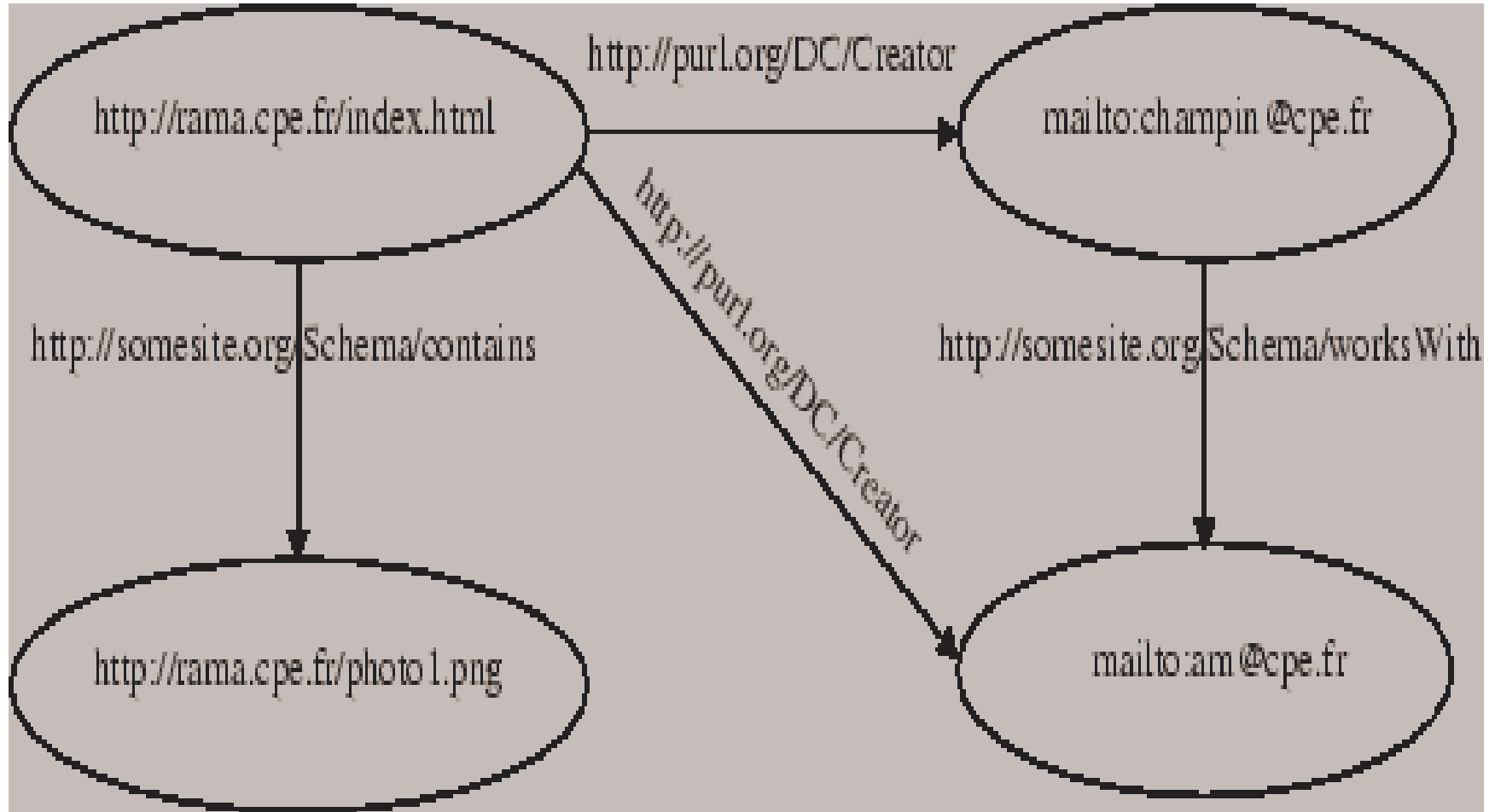


RDF Concepts



- ❑ **Resource:** anything that can have a URI is a resource.
 - RDF uses *qualified URIs*, i.e. URI#local-Id
- ❑ **Property:** A resource that has a name and can be used as a property. Properties are “first class” resources.
- ❑ **Statement:** The base element of the RDF model is the triple: < subject predicate object >
- ❑ **RDF document:** Is a set of triples. Can be represented by a labeled graph. Nodes are qualified URIs. Arcs are also qualified URIs.

RDF graph [10]





RDF Textual representation



□ Triplets

```
<http://rama.cpe.fr/index.html http://purL.org/DC/creator  
mailto:charpin@cpe.fr>
```

```
<http://rama.cpe.fr/index.html http://purL.org/DC/creator  
mailto:am@cpe.fr>
```

```
<mailto:charpin@cpe.fr  
http://somesite.org/Schema/worksWith  
mailto:am@cpe.fr>
```

```
<http://rama.cpe.fr/index.html  
http://somesite.org/Schema/Contains  
http://rama.cpe.fr/photo1.png>
```



RDF Textual representation



□ Triplets

<<http://rama.cpe.fr/index.html> <http://purL.org/DC/creator>
<mailto:charpin@cpe.fr>>

<<http://rama.cpe.fr/index.html> <http://purL.org/DC/creator>
<mailto:am@cpe.fr>>

< <mailto:charpin@cpe.fr>
<http://somesite.org/Schema/worksWith>
<mailto:am@cpe.fr>>

<<http://rama.cpe.fr/index.html>
<http://somesite.org/Schema/Contains>
<http://rama.cpe.fr/photo1.png>>



Define cpe: to stand for “http://rama.cpe.fr/”



□ Triplets

```
<cpe:index.html http://purL.org/DC/creator  
mailto:charpin@cpe.fr>
```

```
<cpe:index.html http://purL.org/DC/creator  
mailto:am@cpe.fr>
```

```
<mailto:charpin@cpe.fr  
http://somesite.org/Schema/worksWith  
mailto:am@cpe.fr>
```

```
<cpe:index.html http://somesite.org/Schema/Contains  
cpe:photo1.png>
```



□ Triplets

<cpe:index.html http://purL.org/DC/creator
mailto:charpin@cpe.fr>

<cpe:index.html http://purL.org/DC/creator
mailto:am@cpe.fr>

<mailto:charpin@cpe.fr
http://somesite.org/Schema/worksWith
mailto:am@cpe.fr>

<cpe:index.html http://somesite.org/Schema/Contains
cpe:photo1.png>



Define dc: to stand for
“http://purL.org/DC/”



□ Triplets

```
<cpe:index.html dc:creator mailto:charpin@cpe.fr>
```

```
<cpe:index.html dc:creator mailto:am@cpe.fr>
```

```
<mailto:charpin@cpe.fr
```

```
http://somesite.org/Schema/worksWith  
mailto:am@cpe.fr>
```

```
<cpe:index.html
```

```
http://somesite.org/Schema/Contains  
cpe:photo1.png>
```



Define sc: to stand for
“<http://somesite.org/Schema/>”



□ Triplets

```
<cpe:index.html dc:creator mailto:charpin@cpe.fr>
```

```
<cpe:index.html dc:creator mailto:am@cpe.fr>
```

```
<mailto:charpin@cpe.fr
```

```
  http://somesite.org/Schema/worksWith
```

```
  mailto:am@cpe.fr>
```

```
<cpe:index.html
```

```
  http://somesite.org/Schema/Contains
```

```
  cpe:photo1.png>
```



□ Triplets

<cpe:index.html dc:creator mailto:charpin@cpe.fr>

<cpe:index.html dc:creator mailto:am@cpe.fr>

<mailto:charpin@cpe.fr sc:worksWith
mailto:am@cpe.fr>

<cpe:index.html sc:Contains cpe:photo1.png>



Resources and Vocabularies



- ❑ If X uses the property “staff” and Y uses property “staff” do they mean the same thing?
 - Yes, if they use the two uses of staff have the same resource URI.
- ❑ The set of URIs used constitute a *vocabulary*.
 - E.g. cd defines the vocabulary {cd:creator, cd:worksWith, ...}
- ❑ RDF supports reuse by supporting vocabulary reuse.
- ❑ RDF supports sharing ... of vocabularies. URI refs from different vocabularies can be mixed freely.



Syntax: RDF/XML



- ❑ Represent the triplets in XML.
- ❑ Need to define a set of XML tags to describe RDF statements.
- ❑ Given a set of RDF statements about resource X,

```
<rdf:RDF>  
<rdf:description rdf:about=X>  
  <property>          </property>  
  <property>          </property>  
</rdf:description>  
</rdf:RDF>
```

Where rdf=<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
Vocabulary RDF, description, about is defined in rdf.



Example written in RDF/XML



```
<?XML version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purL.org/DC/"
  xmlns:sc="http://somesite.org/Schema/">

  <rdf:Description rdf:about="//rama.cpe.fr/index.html">
    <dc:creator rdf:resource="mailto:charpin@cpe.fr"/>
    <dc:creator rdf:resource="mailto:am@cpe.fr"/>
    <sc:Contains rdf:resource="http://rama.cpe.fr/photo1.png"/>
  </rdf:Description>

  <rdf:Description rdf:about="mailto:charpin@cpe.fr">
    <dc:worksWith resource="mailto:am@cpe.fr"/>
  </rdf:Description>
</rdf:RDF>
```

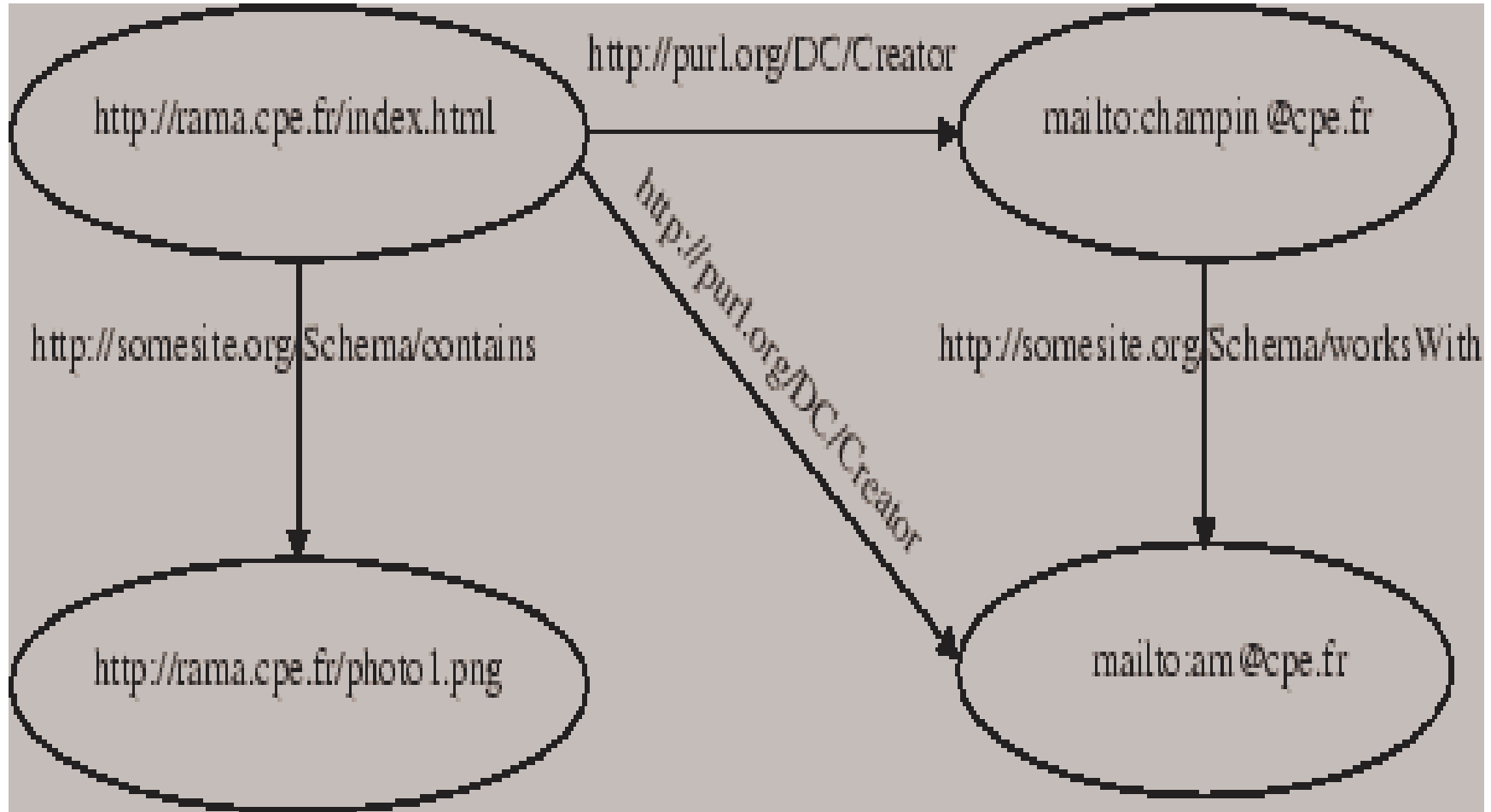


Other features of RDF



- ❑ Use of data types
 - That are defined elsewhere
- ❑ Definition of containers
 - Bags, Sequences, Alt, Collections
- ❑ Reification
 - RDF Statements (triples) are given an id, thus is a resource.
 - RDF statements are made about that resource.
 - Used to capture authorship, date, etc.

RDF graph [10]





What can we conclude from RDF data?



- Champin worksWith Am. Does Am works also with Champin?
- If X worksWith Y and Y worksWith Z, does X worksWith Z?
- Is Champin a person? Do we know that all creators are people?
- What is the nature of the “objects” that Champin can be creator of?
- And many more questions.



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models.
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Reusing Vocabularies and Ontologies
 3. Tools for constructing and maintaining domain model.
4. Summary Conclusion



We use



- XML encoding
- RDF syntax
- RDFS-defined vocabulary**
- OWL-S upper ontology as a basis
- DL as a semantics
- Protégé as the editing tool.



Defining vocabularies



- ❑ More than just registering resource names
- ❑ Describing classes, properties, and their properties. RDF Schema.
- ❑ Classes
 - MotorVehicle rdf:type rdfs:Class
 - Van rdfs:subclass MotorVehicle
 - MiniVan rdfs:subclass Van
 - myVan rdf:type Van



Describing properties



□ Domain and range

- person rdf:type rdfs:class
- woman rdfs:subClassOf Person
- MotherOf rdf:type rdf:Property
- MotherOf rdfs:domain woman
- MotherOf rdfs:range person



Additional information of interest



- Cardinality constraints
- Symmetry, transitivity characteristics of a property
- Equivalence between classes
- Equivalence between properties
- Identity between instances
- Defining classes as expressions (union, intersection, etc.) of other classes



Examples of RDF(S) applications



- ❑ Dublin Core meta data initiative[11] defines vocabulary for documents (title, author, subject, description, publisher, language, coverage, date, type, etc.)
- ❑ PRISM: Publishing Requirements for Industry Standard Metadata [12] builds on the DC core and defines three additional vocabularies PRISM: specializes DC terms (publicationTime, releaseTime, expirationTime,). PVC provides properties for specifying terms in a vocabulary (synonyms, antonyms, etc.). PRL defines the vocabulary for rights management.



Examples of RDF(S) applications



- ❑ RSS 1.0 RDF Site Summary [13]. RDF application for describing, identifying and aggregating dynamic information
- ❑ CIM/XML [14] vocabulary for defining Common Information Model for describing power system entities and relationships
- ❑ Gene Ontology Consortium (GO) [15]



RDF Characteristics



- ❑ Economy of concepts
 - Properties are resources
 - Objects are resources
- ❑ Simple format
 - All statements (subject, property, object)
- ❑ Flexible
 - Anybody can invent their own concepts and properties
- ❑ Sharable
 - Anybody can use properties defined by others
 - Vocabularies are defined
- ❑ Precise
 - Properties of classes and properties can be specified.



RDF-RDFS characteristics



- Allows to specify data types, but does not enforce data type compliance.
- Allows to specify subclass relationships, but does not enforce coherence.
- Allows to specify property characteristics, but does not verify that data entered is consistent with specifications.

Consistency checking, reasoning left for applications using RDF-RDFS



RDF's mission



- Started out as a language to describe resources (data sources).
- Evolved as a language with much more general purpose.



We use



- XML encoding
- RDF syntax
- RDFS-defined vocabulary
- OWL-S upper ontology as a basis
- DL as a semantics
- Protégé as the editing tool.



Ontology of Services



- ❑ RDF emphasized the need for
 - Reusing existing data, vocabularies
 - Sharing new data developed
- ❑ RDF emphasized the importance of the concept of
 - Community
- ❑ Communities share same
 - Domain of interest
 - Vocabulary to talk about the domain
 - Understanding of the key properties of the domain



Tasks community: OWL-S



- Web contents
 - Data, information
 - Services
- “The semantic Web should enable greater access not only to content but also to services on the web.” [16]



Use of web services requires

- Indexing
- Searching
- Comparing
- Composing
- Using

Both information and services can be used by

- Humans
- Other computer agents



OWL-S



- ❑ For computer agents to be able to use services, these services must
 - Be described in a language that is machine interpretable.
 - Be described using a limited set of concepts that are shared by all agents.
 - OWL-S defines the set of concepts on which descriptions are based and
 - OWL-S defines the structure of these descriptions.
 - OWL-S is written in XML-RDF. So descriptions must be written in that language as well.



OWL-S



- ❑ Is built on top RDF/XML
- ❑ and adds
 - Concepts
 - Disjoint
 - Inverse
 - An upper ontology
- ❑ Is built on top of DL which provides
 - Formal semantics
 - Reasoning capabilities



Brief historical



- The Web Ontology Language OWL [17] is a
- ❑ semantic markup language for publishing and sharing ontologies on the World Wide Web.
 - ❑ OWL is a vocabulary extension of RDF
 - ❑ OWL is derived from the DAML+OIL [18] Web Ontology Language.
 - ❑ OWL comes in 3 flavors Lite, DL, Full



OWL Full



- OWL Full allows the free mixing of OWL and RDF.
- Does not enforce strict separation between classes, properties, individuals, and data values.



OWL DL



- OWL Description Logics
- Supports the same language as OWL Full
- Requires classes, properties, individuals, and data values to be disjoint



OWL Lite



- Supports only a subset of OWL DL language constructs



OWL Ontologies



- ❑ OWL documents are ontologies.
- ❑ Ontologies can import other ontologies.
- ❑ OWL document consists of
 - Ontology header
 - Class axioms
 - Property axioms
 - Facts about individuals



OWL Classes



- ❑ Described by one of
 - Class identifier
 - Exhaustive enumeration
 - Property restriction
 - Intersection
 - Union
 - complement



Class Axioms



- subClassOf
- equivalentClass
- disjointWith



Properties



- ❑ Object properties (link individuals to individuals)
- ❑ Datatype properties (link individuals to data values)
- ❑ OWL supports the following property axioms
 - RDF Schema constructs (domain, range, subPropertyOf)
 - Relations to other properties (equivalentProperty, inverseOf)
 - Global cardinality constraints (functionalProperty, inverseFunctionalProperty)
 - Logical property characteristics (SymmetricProperty, TransitiveProperty)



Individuals



- ❑ Most statements are membership statements, e.g. Tosca is an Opera, myCar is a Minivan, etc.
- ❑ Axioms
 - sameAs
 - differentFrom
 - allDifferent



OWL used by



❑ Companies:

- IBM, Boeing, Sun, HP, Network Interface, Adobe Acrobat

❑ Applications

- Bioinformatics
- Fluid Dynamics
- Mathematics



Ontology



□ Three components of ontologies

- Taxonomy hierarchy of concepts
- Internal concept structure and relations between concepts
- Explicit axioms

□ Ontologies differ by

- Extent to which all three components are there
- Top level concepts
- How they handle basic parts: things, processes relations, part-whole relations



□ Ontologies [Gruber]

- An ontology is a specification of a conceptualization.
- An ontology is a description of the concepts and relationships that can exist for an agent or community of agents
- Enabling knowledge sharing and reuse
- Practically, an ontology commitment is an agreement to use a vocabulary in a way that is consistent with respect to the theory specified by the ontology.



□ Ontologies

- Concepts
- Relationships
- Axioms
- Instances

□ Two parts

- Terminological (schema, intentional)
- Factual (data, extensional)



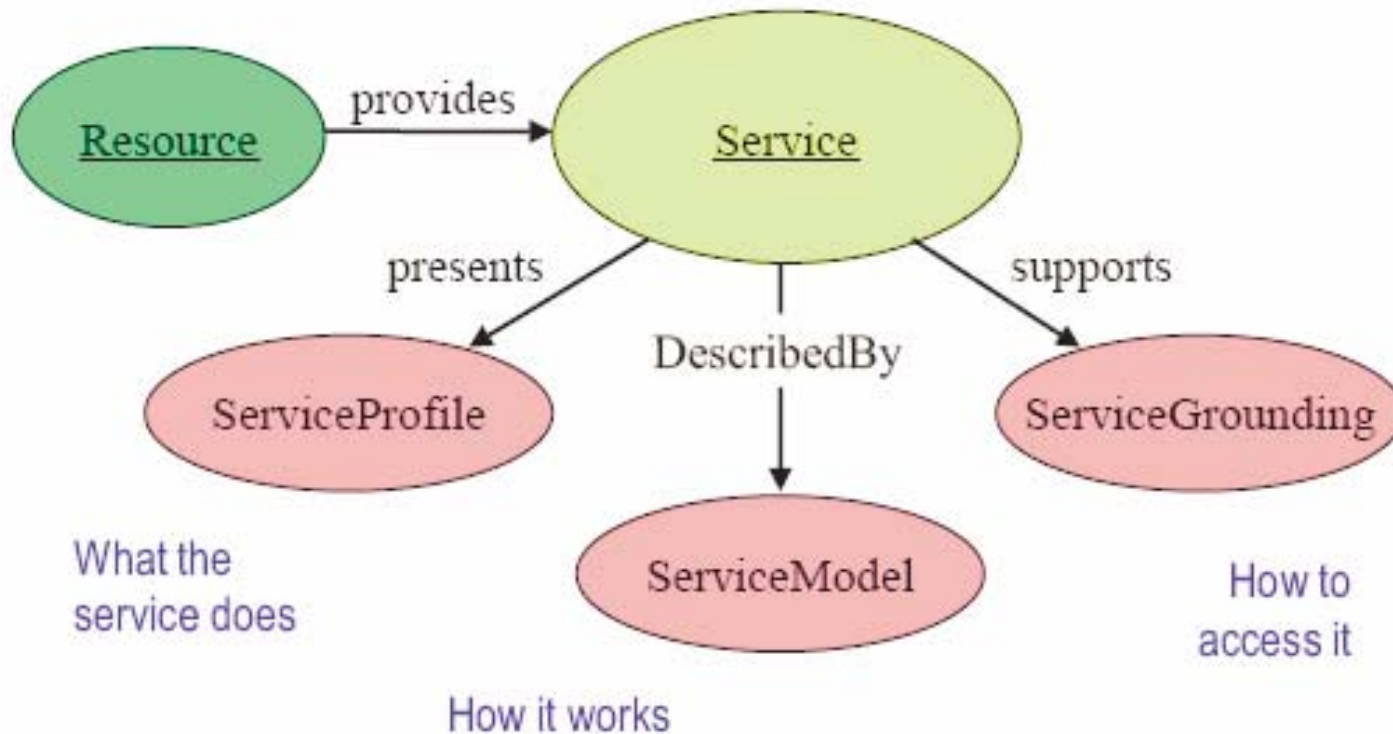
OWL-S: Upper Ontology of Services



❑ Developed to support

- Automatic Web service discovery
- Automatic Web service invocation
- Automatic Web service composition and interoperation

Update this to reflect newest version



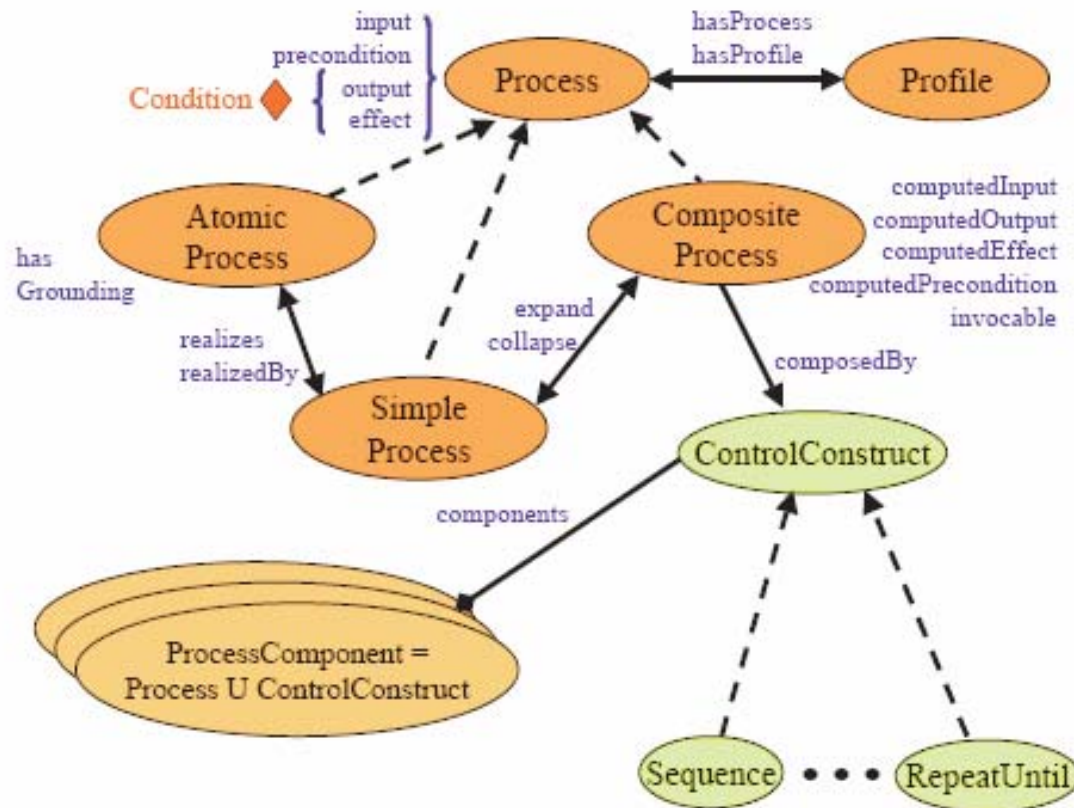


Service Profile Description



- hasInput
- hasOutput
- hasParameters
- hasPrecondition
- hasEffect

Services Modeled as Processes





Composite Process



- Sequence
- Split, Split-Join
- Unordered
- If-Then-Else
- Iterate
- Repeat-Until



Outline



1. Motivation, Requirements
 1. Motivation
 2. Requirements
2. Application view (black box)
 1. Ontology of tasks
 2. Semantics of declarative constructs
 3. Consistency checking and value propagation
 4. Reasoning about domain models.
 5. Mappings from domain model components to performance models
3. Technological view (white box)
 1. Language, representation for capturing the domain model.
 2. Reusing Vocabularies and Ontologies
 3. **Tools for constructing and maintaining domain model.**
4. Summary Conclusion



3. Ontology Editing Tools: Protégé [18]



- Protégé is a free, open source ontology editor and knowledge-base framework.
- Protégé is based on Java, is extensible, and provides a foundation for customized knowledge-based applications.
- Protégé is supported by a strong community of developers and academic, government for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modeling.



Summary, Conclusions



- Performance modeling is a critical component of system design.
- Performance modeling needs to be supported through reuse.
- Domain modeling is a good foundation for reuse in performance modeling.



References



1. Haraldsdottir et al *ATM Preliminary Design Process and Future Operational Concept Evaluation*, Boeing 2000
http://as.nasa.gov/aatt/wspdfs/Haraldsdottir_PD_OpsConEval.pdf
2. NASA Ames Research, *DAG-TM Concept Element 6 En Route Trajectory Negotiation Operational Concept Description*, Feb 12, 2003
http://as.nasa.gov/aatt/rto/RTOFinal72_DAGCE6OCD.pdf.
3. Remington et al Using Apex/ CPM-GOMS to Develop Human-Like Software Agents, Workshop on Human Multi-Agent Systems AAMAS-03 Melbourne, Australia, 2003
http://www.traclabs.com/~cmartin/hmas/wkshp_2003/papers/Remington.pdf
4. Christian Lebiere et al. IMPRINT/ACT-R: Integration of a Task Network Modeling Architecture with a cognitive architecture and its application to human-error modeling. *Proceedings of the Advanced Technologies Simulation Conference*. San Diego, CA., 2002. http://human-factors.arc.nasa.gov/ihi/hcsl/HPM_pubs/Lebiere_SMSI_2002.PDF



5. IMPRINT home page <http://www.arl.army.mil/ARL-Directorates/HRED/imb/imprint/Imprint7.htm>
6. ACT-R home page <http://act-r.psy.cmu.edu/>.
7. W3 Schools XML Tutorial http://www.w3schools.com/xml/xml_what_is.asp
8. W3C PICS The Platform for Content Selection Home Page
<http://www.w3.org/PICS>
9. Eric Miller *An Introduction to the Resource Description Framework*, D-Lib Magazine, 1998, <http://www.dlib.org/dlib/may98/miller/05miller.html>.
10. Pierre-Antoine Champin RDF Tutorial
<http://www710.univ-lyon1.fr/~champin/rdf-tutorial/rdf-tutorial.html>
11. Dublin Core Metadata Element Set, Version 1.1: Reference Description
<http://dublincore.org/documents/2003/06/02/dces/>
12. PRISM Publishing Requirements for Industry Standard Metadata
<http://www.prismstandard.org/>
13. RDF Site Summary (RSS) 1.0 <http://purl.org/rss/1.0/spec>



14. Protégé <http://protege.stanford.edu/>



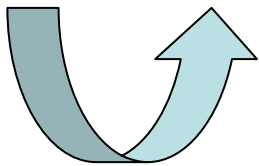
14. Common Information Model (CIM): CIM 10 Version, 2001
[http://www.epri.com/attachements/286161_1001976\(1\).pdf](http://www.epri.com/attachements/286161_1001976(1).pdf)
15. Gene Ontology: tool for the unification of biology, The Gene Ontology Consortium, *Nature Genetics* Vol. 25: 25-29, May 2000.
http://www.geneontology.org/GO_nature_genetics_2000.pdf
16. OWL-S: Semantic Markup for Web Services, the OWL Services Coalition,
<http://www.daml.org/services/>
17. OWL Web Ontology Language Reference <http://www.w3.org/TR/owl-ref/>
18. DAML DARPA Agent Markup Language <http://www.daml.org/>
19. CYC Project <http://www.cyc.com/cyc-21/cover.html>
20. Generalized Upper Model
www.darmstadt.gmd.de/publish/komet/genum/newUM.html
21. M. Matessa et al "Reusable Templates in Human Performance Modeling"
Proceedings of the 24th annual Conference of the Cognitive Science Society, 2002, pp.649-654.
22. Wordnet <ftp://clarity.princeton.edu/pub/wordnet/>



IMPRINT



- Improved Performance Research Integration Tool
- Developed for ARL HRED to conduct human performance analysis early in the acquisition of weapon systems process.
- Hierarchical decomposition of tasks.
- Flow of control between the tasks.
- For more information, see [5].





Recommendations related to the semantic Web



- ❑ XML: surface syntax for structured documents, but imposes no semantic constraints on the meaning of these documents.
- ❑ XML Schema: restricts the structure of XML documents and extends XML datatypes
- ❑ RDF data model for object resources and relations between them, provides a simple semantics
- ❑ RDF Schema vocabulary for describing properties and classes of RDF resources with a semantics for generalization hierarchies of such properties
- ❑ OWL adds more vocabulary for describing properties and classes. In particular, we can specify relations between classes (disjoint), cardinality, equality, richer typing, characteristics of properties, and enumerated classes