Coordination-based Systems

Distributed Systems L-A
Sistemi Distribuiti L-A

Andrea Omicini
andrea.omicini@unibo.it

Ingegneria Due
ALMA MATER STUDIORUM—Università di Bologna a Cesena

Academic Year 2009/2010
Outline

1. Elements of Distributed Systems Engineering
2. Coordination: A Meta-model
3. Enabling vs. Governing Interaction
4. Classifying Coordination Models
Scenarios for Concurrent / Distributed Systems

**Issues**

- **Concurrency / Parallelism**
  - Multiple independent activities / loci of control
  - Active simultaneously
  - Processes, threads, actors, active objects, agents...

- **Distribution**
  - Activities running on different and heterogeneous execution contexts (machines, devices, . . .)

- **“Social” Interaction**
  - Dependencies among activities
  - Collective goals involving activities coordination / cooperation

- **“Environmental” Interaction**
  - Interaction with external resources
  - Interaction within the time-space fabric
Scenarios for Concurrent / Distributed Systems

**Issues**

- **Concurrency / Parallelism**
  - Multiple independent activities / loci of control
  - Active simultaneously
  - Processes, threads, actors, active objects, agents...

- **Distribution**
  - Activities running on different and heterogeneous execution contexts (machines, devices, ...) 

- **“Social” Interaction**
  - Dependencies among activities
  - Collective goals involving activities coordination / cooperation

- **“Environmental” Interaction**
  - Interaction with external resources
  - Interaction within the time-space fabric
**Scenarios for Concurrent / Distributed Systems**

**Issues**

- **Concurrency / Parallelism**
  - Multiple independent activities / loci of control
  - Active simultaneously
  - Processes, threads, actors, active objects, agents...

- **Distribution**
  - Activities running on different and heterogeneous execution contexts (machines, devices, ...)

- **“Social” Interaction**
  - Dependencies among activities
  - Collective goals involving activities coordination / cooperation

- **“Environmental” Interaction**
  - Interaction with external resources
  - Interaction within the time-space fabric
Scenarios for Concurrent / Distributed Systems

**Issues**

- **Concurrency / Parallelism**
  - Multiple independent activities / loci of control
  - Active simultaneously
  - Processes, threads, actors, active objects, agents...

- **Distribution**
  - Activities running on different and heterogeneous execution contexts (machines, devices, ...)

- **“Social” Interaction**
  - Dependencies among activities
  - Collective goals involving activities coordination / cooperation

- **“Environmental” Interaction**
  - Interaction with external resources
  - Interaction within the time-space fabric
Issues

- **Concurrency / Parallelism**
  - Multiple independent activities / loci of control
  - Active simultaneously
  - Processes, threads, actors, active objects, agents...

- **Distribution**
  - Activities running on different and heterogeneous execution contexts
    (machines, devices, ...)

- **“Social” Interaction**
  - Dependencies among activities
  - Collective goals involving activities coordination / cooperation

- **“Environmental” Interaction**
  - Interaction with external resources
  - Interaction within the time-space fabric
Basic Engineering Principles

**Principles**

- **Abstraction**
  - Problems should be faced / represented at the most suitable level of abstraction
  - Resulting “abstractions” should be expressive enough to capture the most relevant problems
  - Conceptual integrity

- **Locality & encapsulation**
  - Design abstractions should embody the solutions corresponding to the domain entities they represent

- **Run-time vs. design-time abstractions**
  - Incremental change / evolutions
  - On-line engineering
  - (Cognitive) Self-organising systems
Basic Engineering Principles

**Principles**

- **Abstraction**
  - Problems should be faced / represented at the most suitable level of abstraction
  - Resulting “abstractions” should be expressive enough to capture the most relevant problems
  - Conceptual integrity

- **Locality & encapsulation**
  - Design abstractions should embody the solutions corresponding to the domain entities they represent

- **Run-time vs. design-time abstractions**
  - Incremental change / evolutions
  - On-line engineering
  - (Cognitive) Self-organising systems
Basic Engineering Principles

**Abstraction**
- Problems should be faced / represented at the most suitable level of abstraction
- Resulting “abstractions” should be expressive enough to capture the most relevant problems
- Conceptual integrity

**Locality & encapsulation**
- Design abstractions should embody the solutions corresponding to the domain entities they represent

**Run-time vs. design-time abstractions**
- Incremental change / evolutions
- On-line engineering
- (Cognitive) Self-organising systems
## Basic Engineering Principles

### Principles

- **Abstraction**
  - Problems should be faced / represented at the most suitable level of abstraction
  - Resulting “abstractions” should be expressive enough to capture the most relevant problems
  - Conceptual integrity

- **Locality & encapsulation**
  - Design abstractions should embody the solutions corresponding to the domain entities they represent

- **Run-time vs. design-time abstractions**
  - Incremental change / evolutions
  - On-line engineering
  - (Cognitive) Self-organising systems
### Open systems
- No hypothesis on the component’s life & behaviour

### Distributed systems
- No hypothesis on the component’s location & motion

### Heterogeneous systems
- No hypothesis on the component’s nature & structure
Which Components?

- **Open systems**
  - No hypothesis on the component’s life & behaviour

- **Distributed systems**
  - No hypothesis on the component’s location & motion

- **Heterogeneous systems**
  - No hypothesis on the component’s nature & structure
### Which Components?

<table>
<thead>
<tr>
<th>Open systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>- No hypothesis on the component’s life &amp; behaviour</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Distributed systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>- No hypothesis on the component’s location &amp; motion</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Heterogeneous systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>- No hypothesis on the component’s nature &amp; structure</td>
</tr>
</tbody>
</table>
Which Interaction? Control vs. Data

How to model an independent activity?

- Objects? No way
  - Objects encapsulate a state and a behaviour, but not a control flow
    - Objects have autonomy over their state, they can control it
    - Objects have not autonomy over their behaviour, they cannot control it
    - Control flows along with data, by means of method invocation (as a reification of message passing)
  - Control is outside objects, owned by human designer who acts as a control authority, establishing the control flow
  - Object interaction is limited and disciplined by interfaces, governed by the human designer

How to model concurrent activities?

- How to model interaction and coordination among concurrent activities?
- How to decouple data and control?
- Method invocation? No way!
Which Interaction? Control vs. Data

How to model an independent activity?

- Objects? No way
  - Objects encapsulate a state and a behaviour, but not a control flow
    - Objects have autonomy over their state, they can control it
    - Objects have not autonomy over their behaviour, they cannot control it
    - Control flows along with data, by means of method invocation (as a reification of message passing)
  - Control is outside objects, owned by human designer who acts as a control authority, establishing the control flow
  - Object interaction is limited and disciplined by interfaces, governed by the human designer

How to model concurrent activities?

- How to model interaction and coordination among concurrent activities?
- How to decouple data and control?
- Method invocation? No way!
Which Interaction? Control vs. Data

How to model an independent activity?

- Objects? No way
- Objects encapsulate a state and a behaviour, but not a control flow
  - Objects have autonomy over their state, they can control it
  - Objects have not autonomy over their behaviour, they cannot control it
  - Control flows along with data, by means of method invocation (as a reification of message passing)
- Control is outside objects, owned by human designer who acts as a control authority, establishing the control flow
- Object interaction is limited and disciplined by interfaces, governed by the human designer

How to model concurrent activities?

- How to model interaction and coordination among concurrent activities?
- How to decouple data and control?
- Method invocation? No way!
The Space of Interaction

interaction space

software component

...
What is a component of an interactive system?

- A computational abstraction characterised by an independent computational activity, and by I/O capabilities
- Independent elaboration / computation and interaction
Algorithmic Computation

Elaboration / Computation

- Turing Machine
- Black box algorithms
- Church and computable functions

Beyond Turing Machines

- Wegner’s Interaction Machines
- Examples: AGV, Chess oracle
Algorithmic Computation

Elaboration / Computation

- Turing Machine
- Black box algorithms
- Church and computable functions

Beyond Turing Machines

- Wegner’s Interaction Machines
- Examples: AGV, Chess oracle
Basics of Interaction

A simple sequential machine

- Output: shows part of its state outside
- Input: bounds a portion of its own state to the outside

Coupling across component’s boundaries

- Information
- Time – internal / sequential vs. external / entropic
Basics of Interaction

A simple sequential machine
- Output: shows part of its state outside
- Input: bounds a portion of its own state to the outside

Coupling across component’s boundaries
- Information
- Time – internal / sequential vs. external / entropic
Compositionality vs. Non-compositionality

**Compositionality**

- Sequential composition $P_1; P_2$

\[ \text{behaviour}(P_1; P_2) = \text{behaviour}(P_1) + \text{behaviour}(P_2) \]

**Non-compositionality**

- Interactive composition $P_1|P_2$

\[ \text{behaviour}(P_1|P_2) = \text{behaviour}(P_1) + \text{behaviour}(P_2) + \text{interaction}(P_1, P_2) \]

Interactive composition is more than the sum of its parts
Compositionality vs. Non-compositionality

### Compositionality
- Sequential composition $P_1; P_2$
- $\text{behaviour}(P_1; P_2) = \text{behaviour}(P_1) + \text{behaviour}(P_2)$

### Non-compositionality
- Interactive composition $P_1|P_2$
- $\text{behaviour}(P_1|P_2) = \text{behaviour}(P_1) + \text{behaviour}(P_2) + \text{interaction}(P_1, P_2)$
- Interactive composition is more than the sum of its parts
Non-compositionality

Issues

- Compositionality vs. formalisability
- Emergent behaviours
- Formalisability vs. expressiveness
Coordination in Distributed Programming

Coordination model as a glue

A coordination model is the glue that binds separate activities into an ensemble [Gelernter and Carriero, 1992]

Coordination model as an agent interaction framework

A coordination model provides a framework in which the interaction of active and independent entities called agents can be expressed [Ciancarini, 1996]

Issues for a coordination model

A coordination model should cover the issues of creation and destruction of agents, communication among agents, and spatial distribution of agents, as well as synchronization and distribution of their actions over time [Ciancarini, 1996]
Coordination in Distributed Programming

Coordination model as a glue

A coordination model is the glue that binds separate activities into an ensemble [Gelernter and Carriero, 1992]

Coordination model as an agent interaction framework

A coordination model provides a framework in which the interaction of active and independent entities called agents can be expressed [Ciancarini, 1996]

Issues for a coordination model

A coordination model should cover the issues of creation and destruction of agents, communication among agents, and spatial distribution of agents, as well as synchronization and distribution of their actions over time [Ciancarini, 1996]
Coordination in Distributed Programming

Coordination model as a glue

A coordination model is the glue that binds separate activities into an ensemble [Gelernter and Carriero, 1992]

Coordination model as an agent interaction framework

A coordination model provides a framework in which the interaction of active and independent entities called agents can be expressed [Ciancarini, 1996]

Issues for a coordination model

A coordination model should cover the issues of creation and destruction of agents, communication among agents, and spatial distribution of agents, as well as synchronization and distribution of their actions over time [Ciancarini, 1996]
Coordination in Distributed Programming

**Coordination model as a glue**

*A coordination model is the glue that binds separate activities into an ensemble* [Gelernter and Carriero, 1992]

**Coordination model as an agent interaction framework**

*A coordination model provides a framework in which the interaction of active and independent entities called agents can be expressed* [Ciancarini, 1996]

**Issues for a coordination model**

*A coordination model should cover the issues of creation and destruction of agents, communication among agents, and spatial distribution of agents, as well as synchronization and distribution of their actions over time* [Ciancarini, 1996]
What is Coordination?

Ruling the space of interaction

coordination

elaboration / computation

...
New Perspective on Computational Systems

Programming languages
- Interaction as an orthogonal dimension
- Languages for interaction / coordination

Software engineering
- Interaction as an independent design dimension
- Coordination patterns

Artificial intelligence
- Interaction as a new source for intelligence
- Social intelligence
New Perspective on Computational Systems

**Programming languages**
- Interaction as an orthogonal dimension
- Languages for interaction / coordination

**Software engineering**
- Interaction as an independent design dimension
- Coordination patterns

**Artificial intelligence**
- Interaction as a new source for intelligence
- Social intelligence
New Perspective on Computational Systems

Programming languages
- Interaction as an orthogonal dimension
- Languages for interaction / coordination

Software engineering
- Interaction as an independent design dimension
- Coordination patterns

Artificial intelligence
- Interaction as a new source for intelligence
- Social intelligence
The medium of coordination

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some coordination laws
  - enacted by the behaviour of the medium
  - defining the semantics of coordination
The *medium of coordination*

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some *coordination laws*
  - enacted by the behaviour of the medium
  - defining the semantics of coordination
The medium of coordination

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some coordination laws
  - enacted by the behaviour of the medium
  - defining the semantics of coordination
The *medium of coordination*

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some *coordination laws*
  - enacted by the behaviour of the medium
  - defining the semantics of coordination
The medium of coordination

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some coordination laws
  - enacted by the behaviour of the medium
  - defining the semantics of coordination
The *medium of coordination*

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some *coordination laws*
  - enacted by the behaviour of the medium
  - defining the semantics of coordination
The medium of coordination

- “fills” the interaction space
- enables / promotes / governs the admissible / desirable / required interactions among the interacting entities
- according to some coordination laws
  - enacted by the behaviour of the medium
  - defining the semantics of coordination
A constructive approach

Which are the components of a coordination system?
- **Coordination entities**: Entities whose mutual interaction is ruled by the model, also called the *coordinables*.
- **Coordination media**: Abstractions enabling and ruling agent interactions.
- **Coordination laws**: Rules defining the behaviour of the coordination media in response to interaction.
A constructive approach

Which are the components of a coordination system?

Coordination entities Entities whose mutual interaction is ruled by the model, also called the *coordinables*

Coordination media Abstractions enabling and ruling agent interactions

Coordination laws Rules defining the behaviour of the coordination media in response to interaction
A constructive approach

Which are the components of a coordination system?

**Coordination entities**  Entities whose mutual interaction is ruled by the model, also called the *coordinables*

Coordination media  Abstractions enabling and ruling agent interactions

Coordination laws  Rules defining the behaviour of the coordination media in response to interaction
A constructive approach

Which are the components of a coordination system?

**Coordination entities** Entities whose mutual interaction is ruled by the model, also called the *coordinables*

**Coordination media** Abstractions enabling and ruling agent interactions

**Coordination laws** Rules defining the behaviour of the coordination media in response to interaction
A constructive approach

Which are the components of a coordination system?

**Coordination entities**  Entities whose mutual interaction is ruled by the model, also called the *coordinables*

**Coordination media**  Abstractions enabling and ruling agent interactions

**Coordination laws**  Rules defining the behaviour of the coordination media in response to interaction
These are the entity types that are coordinated. These could be Unix-like processes, threads, concurrent objects and the like, and even users.

examples Processes, threads, objects, human users, agents, . . .

focus Observable behaviour of the coordinables

question Are we anyhow concerned here with the internal machinery / functioning of the coordinable, in principle?

→ This issue will be clear when comparing Linda & TuCSoN agents
Coordinables

Original definition [Ciancarini, 1996]

These are the entity types that are coordinated. These could be Unix-like processes, threads, concurrent objects and the like, and even users.

**examples** Processes, threads, objects, human users, agents, ...

**focus** Observable behaviour of the coordinables

**question** Are we anyhow concerned here with the internal machinery / functioning of the coordinable, in principle?

→ This issue will be clear when comparing Linda & TuCSoN agents
Coordinables

Original definition [Ciancarini, 1996]

*These are the entity types that are coordinated. These could be Unix-like processes, threads, concurrent objects and the like, and even users.*

**examples** Processes, threads, objects, human users, agents, . . .

**focus** Observable behaviour of the coordinables

**question** Are we anyhow concerned here with the internal machinery / functioning of the coordinable, in principle?

→ This issue will be clear when comparing Linda & TuCSoN agents
Coordinables

**Original definition [Ciancarini, 1996]**

*These are the entity types that are coordinated. These could be Unix-like processes, threads, concurrent objects and the like, and even users.*

**Examples**
- Processes, threads, objects, human users, agents, ...

**Focus**
- Observable behaviour of the coordinables

**Question**
- Are we anyhow concerned here with the internal machinery / functioning of the coordinable, in principle?
  
  → This issue will be clear when comparing Linda & TuCSoN agents
Coordinables

Original definition [Ciancarini, 1996]

*These are the entity types that are coordinated. These could be Unix-like processes, threads, concurrent objects and the like, and even users.*

**examples** Processes, threads, objects, human users, agents, ...  
**focus** Observable behaviour of the coordinables  
**question** Are we anyhow concerned here with the internal machinery / functioning of the coordinable, in principle?  
→ This issue will be clear when comparing Linda & TuCSoN agents
Coordination media

Original definition [Ciancarini, 1996]

*These are the media making communication among the agents possible. Moreover, a coordination medium can serve to aggregate agents that should be manipulated as a whole. Examples are classic media such as semaphores, monitors, or channels, or more complex media such as tuple spaces, blackboards, pipelines, and the like.*

examples  Semaphors, monitors, channels, tuple spaces, blackboards, pipes, . . .
focus  The core around which the components of the system are organised
question  Which are the possible computational models for coordination media?
  →  This issue will be clear when comparing Linda tuple spaces & ReSpecT tuple centres
Coordination media

Original definition [Ciancarini, 1996]

These are the media making communication among the agents possible. Moreover, a coordination medium can serve to aggregate agents that should be manipulated as a whole. Examples are classic media such as semaphores, monitors, or channels, or more complex media such as tuple spaces, blackboards, pipelines, and the like.

examples  Semaphors, monitors, channels, tuple spaces, blackboards, pipes, . . .

focus  The core around which the components of the system are organised

question  Which are the possible computational models for coordination media?

→  This issue will be clear when comparing Linda tuple spaces & ReSpecT tuple centres
### Coordination media

**Original definition [Ciancarini, 1996]**

These are the media making communication among the agents possible. Moreover, a coordination medium can serve to aggregate agents that should be manipulated as a whole. Examples are classic media such as semaphores, monitors, or channels, or more complex media such as tuple spaces, blackboards, pipelines, and the like.

| examples | Semaphors, monitors, channels, tuple spaces, blackboards, pipes, ... |
| focus    | The core around which the components of the system are organised    |
| question | Which are the possible computational models for coordination media? |
|          | → This issue will be clear when comparing Linda tuple spaces & ReSpecT tuple centres |
Coordination media

Original definition [Ciancarini, 1996]

*These are the media making communication among the agents possible. Moreover, a coordination medium can serve to aggregate agents that should be manipulated as a whole. Examples are classic media such as semaphores, monitors, or channels, or more complex media such as tuple spaces, blackboards, pipelines, and the like.*

**examples** Semaphors, monitors, channels, tuple spaces, blackboards, pipes, . . .

**focus** The core around which the components of the system are organised

**question** Which are the possible computational models for coordination media?

→ This issue will be clear when comparing Linda tuple spaces & ReSpecT tuple centres
Coordination media

Original definition [Ciancarini, 1996]

These are the media making communication among the agents possible. Moreover, a coordination medium can serve to aggregate agents that should be manipulated as a whole. Examples are classic media such as semaphores, monitors, or channels, or more complex media such as tuple spaces, blackboards, pipelines, and the like.

examples  Semaphors, monitors, channels, tuple spaces, blackboards, pipes, . . .
focus      The core around which the components of the system are organised
question   Which are the possible computational models for coordination media?
           → This issue will be clear when comparing Linda tuple spaces & ReSpecT tuple centres
Coordination laws

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behaviour of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the communication language, as the syntax used to express and exchange data structures examples tuples, XML elements, FOL terms, (Java) objects
  - the coordination language, as the set of the asmissible interaction primitives, along with their semantics examples in/out/rd (Linda), send/receive (channels), push/pull (pipes)
Coordination laws

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behaviour of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the communication language, as the syntax used to express and exchange data structures examples: tuples, XML elements, FOL terms, (Java) objects
  - the coordination language, as the set of the admissible interaction primitives, along with their semantics examples: in/out/rd (Linda), send/receive (channels), push/pull (pipes)
Coordination laws

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behavior of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the communication language, as the syntax used to express and exchange data structures
    examples: tuples, XML elements, FOL terms, (Java) objects
  - the coordination language, as the set of the admissible interaction primitives, along with their semantics
    examples: in/out/rd (Linda), send/receive (channels), push/pull (pipes)
Coordination laws

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behaviour of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the communication language, as the syntax used to express and exchange data structures. Examples: tuples, XML elements, FOL terms, (Java) objects, ...
  - the coordination language, as the set of the asmissible interaction primitives, along with their semantics. Examples: in/out/rd (Linda), send/receive (channels), push/pull (pipes), ...
### Coordination laws

**Original definition [Ciancarini, 1996]**

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behaviour of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the *communication language*, as the syntax used to express and exchange data structures
    - examples: tuples, XML elements, FOL terms, (Java) objects, …
  - the *coordination language*, as the set of the admissible interaction primitives, along with their semantics
    - examples: in/out/rd (Linda), send/receive (channels), push/pull (pipes), …
Coordination laws

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behaviour of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the communication language, as the syntax used to express and exchange data structures
    examples tuples, XML elements, FOL terms, (Java) objects, ...
  - the coordination language, as the set of the asmissible interaction primitives, along with their semantics
    examples in/out/rd (Linda), send/receive (channels), push/pull (pipes), ...
Coordination laws

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behaviour of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the communication language, as the syntax used to express and exchange data structures
    - examples tuples, XML elements, FOL terms, (Java) objects, ...
  - the coordination language, as the set of the admissible interaction primitives, along with their semantics
    - examples in/out/rd (Linda), send/receive (channels), push/pull (pipes), ...
Coordination laws

Original definition [Ciancarini, 1996]

A coordination model should dictate a number of laws to describe how agents coordinate themselves through the given coordination media and using a number of coordination primitives. Examples are laws that enact either synchronous or asynchronous behaviors or exploit explicit or implicit naming schemes for coordination entities.

- Coordination laws define the behaviour of the coordination media in response to interaction
  - a notion of (admissible interaction) event is required to define a model
- Coordination laws are expressed in terms of
  - the communication language, as the syntax used to express and exchange data structures
    examples tuples, XML elements, FOL terms, (Java) objects, ...
  - the coordination language, as the set of the asmissible interaction primitives, along with their semantics
    examples in/out/rd (Linda), send/receive (channels), push/pull (pipes), ...
What Do We Ask to a Coordination Model?

- to provide high-level abstractions and powerful mechanisms for distributed system engineering
- to enable and promote the construction of open, distributed, heterogeneous systems
- to intrinsically add properties to systems independently of components
  - e.g. flexibility, control, intelligence, ...
Examples of Coordination Mechanisms I

Message passing

- communication among peers
- no abstractions apart from message
- no limitations
  - the notion of protocol could be added as a coordination abstraction
- no intrinsic model of coordination
- any pattern of coordination can be superimposed – again, protocols
Examples of Coordination Mechanisms II

Agent Communication Languages

- **Goal:** promote information exchange
- **Examples:** Arcol, KQML
- **Standard:** FIPA ACL
- **Semantics:** ontologies
- **Enabling communication**
  - ACLs *create* the space of inter-agent communication
  - they do not allow to *constrain* it
- No coordination, again, if not with protocols
Examples of Coordination Mechanisms III

Service-Oriented Architectures

- Basic abstraction: service
- Basic pattern: Service request / response
- Several standards
- Very simple pattern of coordination
Web Server

- Basic abstraction: resource (REST/ROA)
- Basic pattern: Resource request / representation / response
- Several standards
- Again, a very simple pattern of coordination
- Generally speaking, objects, HTTP, applets, JavaScript with AJAX, user interface
  - a multi-coordinated systems
  - “spaghetti-coordination”, no value added from composition
- How can we “fill” the space of interaction to add value to systems?
  - so, how do we get value from coordination?
Examples of Coordination Mechanisms V

Middleware

- Goal: to provide global properties across distributed systems
- Idea: fill the space of interaction with abstractions and shared features
  - interoperability, security, transactionality, ...
- Middleware can contain coordination abstractions
  - but, it can contain anything, so we need to look at specific middleware
CORBA

- Goal: managing object interaction across a distributed systems in a transparent way
- Key features: ORB, IDL, CORBAServices...
- However, no model for coordination
  - just the client-servant pattern
- However, it can provide a shared support for any coordination abstraction or pattern
Enabling interaction

- ACL, middleware, mediators...
- enabling communication
- enabling components interoperation
- no models for coordination of components
  - no rules on what components should (not) say and do at any given moment, depending on what other components say and do, and on what happens inside and outside the system
Governing interaction

- ruling communication
- providing concepts, abstractions, models, mechanisms for meaningful component integration
- governing mutual component interaction, and environment-component interaction
- in general, a model that does
  - rule what components should (not) say and do at any given moment
  - depending on what other components say and do, and on what happens inside and outside the system
## Two Classes for Coordination Models

<table>
<thead>
<tr>
<th>Control-oriented vs. Data-oriented Models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Control-oriented</strong> Focus on the <em>acts</em> of communication</td>
</tr>
<tr>
<td><strong>Data-oriented</strong> Focus on the <em>information</em> exchanged during communication</td>
</tr>
<tr>
<td>— Several surveys, no time enough here</td>
</tr>
<tr>
<td>— Are these really <em>classes</em>?</td>
</tr>
<tr>
<td>— actually, better to take this as a criterion to observe coordination models, rather than to separate them</td>
</tr>
</tbody>
</table>

---

[Control-oriented vs. Data-oriented Models](#)

- Control-driven vs. Data-driven Models
  - [Papadopoulos and Arbab, 1998]
### Processes as black boxes
- I/O ports
- events & signals on state

### Coordinators...
- ...create coordinated processes as well as communication channels
- ...determine and change the topology of communication
- Hierarchies of coordinables / coordinators are possible
Coordinators as meta-level communication components

General features
- High flexibility, high control
- Separation between communication / coordination and computation / elaboration

Examples
- RAPIDE
- Manifold
- ConCoord
- Reo
Main features

- coordinators
- control-driven evolution
  - events without parameters
- stateful communication
- coordination via topology
- fine-grained coordination
- typical example: sort-merge
Control-oriented Models: Impact on Design

Which abstractions?

- Producer-consumer pattern
- Point-to-point communication
- Coordinator
- Coordination as configuration of topology

Which systems?

- Fine-grained granularity
- Fine-tuned control
- Good for small-scale, closed systems
Control-oriented Models: Impact on Design

Which abstractions?
- Producer-consumer pattern
- Point-to-point communication
- Coordinator
- Coordination as configuration of topology

Which systems?
- Fine-grained granularity
- Fine-tuned control
- Good for small-scale, closed systems
Classifying Coordination Models

An Evolutionary Pattern?

Paradigms of sequential programming
- Imperative programming with “goto”
- Structured programming (procedure-oriented)
- Object-oriented programming (data-oriented)

Paradigms of coordination programming
- Message-passing coordination
- Control-oriented coordination
- Data-oriented coordination
An Evolutionary Pattern?

Paradigms of sequential programming
- Imperative programming with “goto”
- Structured programming (procedure-oriented)
- Object-oriented programming (data-oriented)

Paradigms of coordination programming
- Message-passing coordination
- Control-oriented coordination
- Data-oriented coordination
### Data-oriented Models I

#### Communication channel
- Shared memory abstraction
- Stateful channel

#### Processes
- Emitting / receiving data / information

#### Coordination
- Access / change / synchronise on shared data
Data-oriented Models II

Shared dataspace: constraint on communication

shared dataspace
Data-oriented Models

General features

- Expressive communication abstraction
  → information-based design
- Possible spatio-temporal uncoupling
- No control means no flexibility??
- Examples
  - Gamma / Chemical coordination
  - Linda & friends / tuple-based coordination
Summing Up

Coordination for Distributed System Engineering
- Engineering the space of interaction among components

Coordination as Governing Interaction
- Enabling vs. Governing

Classes and Features of Coordination Models
- Control-oriented vs. Data-oriented Models
Conclusions

Summing Up

Coordination for Distributed System Engineering
- Engineering the space of interaction among components

Coordination as Governing Interaction
- Enabling vs. Governing

Classes and Features of Coordination Models
- Control-oriented vs. Data-oriented Models
Summing Up

Coordination for Distributed System Engineering
- Engineering the space of interaction among components

Coordination as Governing Interaction
- Enabling vs. Governing

Classes and Features of Coordination Models
- Control-oriented vs. Data-oriented Models
Elements of Distributed Systems Engineering

Coordination: A Meta-model

Enabling vs. Governing Interaction

Classifying Coordination Models
Bibliography I

Coordination models and languages as software integrators.

Coordination languages and their significance.

Coordination models and languages.
Coordination-based Systems

Distributed Systems L-A
Sistemi Distribuiti L-A

Andrea Omicini
andrea.omicini@unibo.it

Ingegneria Due
Alma Mater Studiorum—Università di Bologna a Cesena

Academic Year 2009/2010