Specifying Data-Flow Requirements for the Automated Composition of Web Services

Annapaola Marconi
marconi@itc.it

April 27, 2006 - ASTRO Seminar
1 Introduction
   • Aim

2 Data Flow Language
   • Syntax
   • VTA Case Study
   • Semantics

3 Automated Composition Framework
Specify how messages sent to component services must be obtained from messages received from component services. 

- how several input messages must be combined to obtain an output message,
- whether an input message can be used several times or just once
- whether all messages received must be processed and sent or not
Language Aim

- Specify how messages sent to component services must be obtained from messages received from component services. e.g.:
  - how several input messages must be combined to obtain an output message,
  - whether an input message can be used several times or just once
  - whether all messages received must be processed and sent or not

- Specify message precedences and other requirements related to the order in which messages must be sent is not an issue of this language.

Annapaola Marconi marconi@itc.it
Specifying Data-Flow Requirements for the Automated Composition of Web Services
Outline

1 Introduction
   - Aim

2 Data Flow Language
   - Syntax
   - VTA Case Study
   - Semantics

3 Automated Composition Framework
A connection node can be external or internal. An external connection node is associated to an output (or an input) external port. Intuitively, an external input (output) node characterizes an external source (target) of data and it is used to exchange data with the outside world.
Identity

It is connected to one connection node in input and one node in output.

The graphical notation for the data-flow identity element \( id(a)(b) \), with input node \( a \) and output node \( b \), is the following:
Identity

It is connected to one connection node in input and one node in output.

The graphical notation for the data-flow identity element \(\text{id}(a)(b)\), with input node \(a\) and output node \(b\), is the following:

Intuitive Semantics: it forwards data received from the input node to the output node.
Operation

It is related to a function definition, is connected to as many input nodes as the number of function parameters and only to one output node corresponding to the function result.

The graphical notation for the data-flow operation element \( \text{oper}[f](a,b)(c) \) characterizing function \( f \), with input nodes \( a \) and \( b \) and output node \( c \), is the following:

![Graphical notation for operation](image)
Operation

It is related to a function definition, is connected to as many input nodes as the number of function parameters and only to one output node corresponding to the function result.

The graphical notation for the data-flow operation element \( \text{oper}[f](a,b)(c) \) characterizing function \( f \), with input nodes \( a \) and \( b \) and output node \( c \), is the following:

![Diagram of a function with input nodes a and b, and an output node c]

**Intuitive Semantics:** when it receives data from all the input nodes, it computes the result and forwards it to the output channel.
Fork

It is connected to a node in input and to as many nodes as necessary in output.

The graphical notation for the data-flow fork element $\text{fork}(a)(b,c)$, with input node $a$ and output nodes $b$ and $c$, is the following:
Fork

It is connected to a node in input and to as many nodes as necessary in output.

The graphical notation for the data-flow fork element \texttt{fork(a)(b,c)}, with input node \texttt{a} and output nodes \texttt{b} and \texttt{c}, is the following:

\begin{center}
\begin{tikzpicture}
  \node (a) at (0,0) [input, fill=black] {a};
  \node (b) at (1,1) [output, fill=black] {b};
  \node (c) at (1,-1) [output, fill=black] {c};
  \draw [->] (a) -- (b);
  \draw [->] (a) -- (c);
\end{tikzpicture}
\end{center}

\textbf{Intuitive Semantics:} It forwards data received on the input node to all the output nodes.
Language Components

Merge

It is connected to one node in output and as many nodes as necessary in input.

We represent the data-flow merge element \( \text{merge}(a,b)(c) \), with input nodes \( a \) and \( b \) and output node \( c \) as:

![Diagram of merge element](attachment:image.png)

Intuitive Semantics:
It forwards data received on some input node to the output node. It preserves the temporal order of data arriving on input nodes (if it receives data on two or more input nodes at the same time, the order is nondeterministic).

Annapaola Marconi
marconi@itc.it

Specifying Data-Flow Requirements for the Automated Composition of Web Services
Merge

It is connected to one node in output and as many nodes as necessary in input.

We represent the data-flow merge element $\text{merge}(a, b)(c)$, with input nodes $a$ and $b$ and output node $c$ as:

![Diagram of Merge](image)

Intuitive Semantics: It forwards data received on some input node to the output node. It preserves the temporal order of data arriving on input nodes (if it receives data on two or more input nodes at the same time, the order is nondeterministic).
**Cloner**

It is connected to one node in input and one node in output.

The data-flow cloner element `clone(a)(b)`, with input node `a` and output node `b` is represented as:

![Diagram of cloner](image)

Intuitive Semantics: It forwards, one or more times, data received from the input node to the output node.
Cloner

It is connected to one node in input and one node in output.

The data-flow cloner element \( \text{clone}(a)(b) \), with input node \( a \) and output node \( b \) is represented as:

\[
\begin{array}{c}
\bullet \\
a \\
+ \\
b \\
\bullet
\end{array}
\]

**Intuitive Semantics:** It forwards, one or more times, data received from the input node to the output node.
Language Components

Filter

It is connected to one node in input and one node in output.

We represent the data-flow filter element \texttt{filt(a)(b)}, having input node \texttt{a} and output node \texttt{b} as:

\begin{center}
\begin{tikzpicture}
    \node (a) at (0,0) [shape=circle,draw] {a};
    \node (b) at (1,0) [shape=circle,draw] {b};
    \node (c) [shape=circle,draw,fill=white,minimum size=0.5cm] at (0.5,0) {?};
    \draw[->] (a) -- (c);
    \draw[->] (c) -- (b);
\end{tikzpicture}
\end{center}
Filter

It is connected to one node in input and one node in output.

We represent the data-flow filter element $\text{filt}(a)(b)$, having input node $a$ and output node $b$ as:

```
  a  ?  b
```

**Intuitive Semantics:** When it receives data on the input node it either forwards it to the output node or discards it.
Last

It is connected to one node in input and one node in output.

The graphical notation for the data-flow last element $\text{last}(a)(b)$, with input node $a$ and output node $b$, is the following:

\[ a \rightarrow \text{L} \rightarrow b \]
Last

It is connected to one node in input and one node in output.

The graphical notation for the data-flow last element \texttt{last(a)(b)}, with input node \texttt{a} and output node \texttt{b}, is the following:

\[ \text{a} \quad \text{L} \quad \text{b} \]

**Intuitive Semantics:** It forwards to the output node at most one data: the last data received on the input node. It discards all other data received.
The diagram obtained by modeling the data flow requirements with the specified language is called **data net**.
The diagram obtained by modeling the data flow requirements with the specified language is called data net.

A data net $\mathcal{D}$ is characterized by:

- a set of external connection nodes associated to input ports $N_{ext}^I$,
- a set of external connection nodes associated to output ports $N_{ext}^O$,
- a set of internal connection nodes $N_{int}$,
- a set of data-flow elements $D$
- and a set of data values $V$. 
Outline

1. Introduction
   - Aim

2. Data Flow Language
   - Syntax
   - VTA Case Study
   - Semantics

3. Automated Composition Framework
The VTA Case Study

Flight Service Protocol

- Request (date, loc)
- Check Avail (date, loc)
- !not_avail
- Offer (cost, schedule)
- ?ack
- ?cancel
- booked(info)
- Not Avail
- Canceled
- Booked

Annapaola Marconi   marconi@itc.it
The VTA Case Study

Flight Service Protocol

**Flight Input Ports**
- f.request.date
- f.request.loc

**Flight Output Ports**
- f.offer.cost
- f.offer.schedule
- f.booked.info
The VTA Case Study

Hotel Service Protocol

Outline
- Introduction
- Data Flow Language
- Automated Composition Framework

Syntax
- VTA Case Study
- Semantics

Annapaola Marconi  marconi@itc.it

Specifying Data-Flow Requirements for the Automated Composition of Web Services

Annapaola Marconi  marconi@itc.it

Data Flow Language

Automated Composition Framework

Syntax

VTA Case Study

Semantics

The VTA Case Study

Hotel Service Protocol

START

? request(date,loc)

check_avail(date,loc)

! not_avail

? cancel

? other

not_avail

! offer(cost, hotel_info)

? cancel

? other

in_avail

CANCELED

booked

BOOLED

Semantics

VTA Case Study

Syntax

Data Flow Language

Automated Composition Framework

Outline

Introduction

Annapaola Marconi  marconi@itc.it

Specifying Data-Flow Requirements for the Automated Composition of Web Services
The VTA Case Study

Hotel Service Protocol

Hotel Input Ports
- h.request.date
- h.request.loc

Hotel Output Ports
- h.offer.cost
- h.offer.hotel_info
The VTA Case Study

AllMaps Service Protocol

```
START

? request(from_loc, to_loc)

! info(map)

SUCC
```
The VTA Case Study

AllMaps Service Protocol

START

? request(from_loc, to_loc)

! info(map)

Succ

AllMaps Input Ports
m.request.from_loc
m.request.to_loc

AllMaps Output Ports
m.info.map
The VTA Case Study

Customer Service Protocol

![Diagram of Customer Service Protocol]

- Customer Input Ports:
  - offer.cost
  - offer.schedule
  - offer.cost
  - offer.hotel
  - offer.dist

- Customer Output Ports:
  - request.date
  - request.loc

Annapaola Marconi
marconi@itc.it

Specifying Data-Flow Requirements for the Automated Composition of Web Services
Specifying Data-Flow Requirements for the Automated Composition of Web Services

Customer Service Protocol

Customer Input Ports
- c.f_offer.cost
- c.f_offer.schedule
- c.h_offer.cost
- c.f_offer.hotel
- c.f_offer.dist_map
- c.booked.cost
- c.booked.info

Customer Output Ports
- c.request.date
- c.request.loc
The VTA Case Study

VTA datareq 1

C.request.date \rightarrow ! \rightarrow F.request.date
The VTA Case Study

VTA datareq 2

C.request.loc \rightarrow \bullet \rightarrow \bullet F.request.loc

\rightarrow \bullet H.request.loc
The VTA Case Study

VTA datareq 3

H.offer.cost \rightarrow \rightarrow \bullet C.h_offer.cost

F.offer.cost \rightarrow \bullet \rightarrow L \rightarrow \bullet prep_cost \rightarrow C.booked.cost

\rightarrow \bullet C.f_offer.cost
The VTA Case Study

VTA datareq 4

```plaintext
F.offer.schedule → get_airport
      ^
      ↓
      get_date

H.request.date

C.f_offer.schedule

M.request.from_loc
```

Annapaola Marconi  marconi@itc.it
### The VTA Case Study

#### VTA data net

- **C.request.date** → ! → F.request.date
- **C.request.loc** → ◼ → F.request.loc
- **H.offer.cost** → ◼ → C.h_offer.cost
- **F.offer.cost** → ◼ → C.booked.cost → C.f_offer.cost
- **F.offer.schedule** → ◼ → C.f_offer.schedule
- **H.offer.info** → ◼ → C.h_offer.hotel
- **F.booked.info** → ◼ → C.booked.info
- **M.info.map** → ◼ → C.h_offer.dist_map

---

**Annapaola Marconi**  
marconi@itc.it  

**Specifying Data-Flow Requirements for the Automated Composition of Web Services**
1. Introduction
   - Aim

2. Data Flow Language
   - Syntax
   - VTA Case Study
   - Semantics

3. Automated Composition Framework
Some definitions..

Given a data net $\mathcal{D} = \langle N^I_{\text{ext}}, N^O_{\text{ext}}, N_{\text{int}}, D, V \rangle$, 

$\text{given an execution } \rho \text{ it's projection on a set of connection nodes, denoted it with } \Pi_{\mathcal{N}}(\rho), \text{ is the ordered sequence } e'_0, .., e'_m \text{ representing the events in } \rho \text{ on a node in } \mathcal{N}$. 
Some definitions..

Given a data net $\mathcal{D} = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$,

- $N_{ext} = N^I_{ext} \cup N^O_{ext}$ is the set of all external connection nodes.
Some definitions..

Given a data net $\mathcal{D} = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$,

- $N_{ext} = N^I_{ext} \cup N^O_{ext}$ is the set of all external connection nodes.
- an event $e$ is a couple $\langle n, v \rangle$, where $n \in N_{ext} \cup N_{int}$, and $v \in V$, which models the fact that the data value $v$ passes through the connection node $n$. 

Annapaola Marconi
marconi@itc.it
Specifying Data-Flow Requirements for the Automated Composition of Web Services
Some definitions..

Given a data net $D = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$,

- $N_{ext} = N^I_{ext} \cup N^O_{ext}$ is the set of all external connection nodes.
- an event $e$ is a couple $\langle n, v \rangle$, where $n \in N_{ext} \cup N_{int}$, and $v \in V$, which models the fact that the data value $v$ passes through the connection node $n$.
- an execution $\rho$ is a finite sequence of events $e_0, .. e_n$. 

Annapaola Marconi
marconi@itc.it

Specifying Data-Flow Requirements for the Automated Composition of Web Services
Some definitions...

Given a data net \( D = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle \),

- \( N_{ext} = N^I_{ext} \cup N^O_{ext} \) is the set of all external connection nodes.
- an event \( e \) is a couple \( \langle n, v \rangle \), where \( n \in N_{ext} \cup N_{int} \), and \( v \in V \), which models the fact that the data value \( v \) passes through the connection node \( n \).
- an execution \( \rho \) is a finite sequence of events \( e_0, .. e_n \).
- given an execution \( \rho \) it’s projection on a set of connection nodes \( N \), denoted it with \( \Pi_N(\rho) \), is the ordered sequence \( e'_0, .. e'_m \) representing the events in \( \rho \) on a node in \( N \).
Accepting execution

An execution $\rho$ is accepted by a data net $\mathcal{D} = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$ if it satisfies all the following properties:
Accepting execution

An execution $\rho$ is accepted by a data net $D = \langle N^l_{ext}, N^o_{ext}, N_{int}, D, V \rangle$ if it satisfies all the following properties:

Property 1

for each identity element $id(a)(b)$ in $D$:

$$\Pi_{\{a,b\}}(\rho) = \left( \sum_{v \in V} \langle a, v \rangle \cdot \langle b, v \rangle \right)^*$$
Accepting execution

An execution $\rho$ is accepted by a data net $D = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$ if it satisfies all the following properties:

**Property 2**

for each operation element $\text{oper}[f](a,b)(c)$ in $D$:

$$\Pi_{\{a,b,c\}}(\rho) = \left( \sum_{v,w \in V} (\langle a, v \rangle \cdot \langle b, w \rangle + \langle b, w \rangle \cdot \langle a, v \rangle) \cdot \langle c, f(v, w) \rangle \right)^* \cdot \sum_{v,w \in V} (\langle a, v \rangle + \langle b, w \rangle + \epsilon)$$
Accepting execution

An execution $\rho$ is accepted by a data net $\mathcal{D} = \langle N^I_{\text{ext}}, N^O_{\text{ext}}, N_{\text{int}}, D, V \rangle$ if it satisfies all the following properties:

Property 3

for each fork element $\text{fork}(a)(b,c)$ in $\mathcal{D}$:

$$\Pi_{\{a,b,c\}}(\rho) = \left( \sum_{v \in V} \langle a, v \rangle \cdot (\langle b, v \rangle \cdot \langle c, v \rangle + \langle c, v \rangle \cdot \langle b, v \rangle) \right)^*$$
Accepting execution

An execution $\rho$ is accepted by a data net $D = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$ if it satisfies all the following properties:

Property 4

for each merge element $\text{merge}(a, b)(c)$ in $D$:

$$\Pi_{\{a,b,c\}}(\rho) = \left( \sum_{v \in V} (\langle a, v \rangle \cdot \langle c, v \rangle + \langle b, v \rangle \cdot \langle c, v \rangle) \right)^*$$
Accepting execution

An execution $\rho$ is accepted by a data net $D = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$ if it satisfies all the following properties:

**Property 5**

For each cloner element $\text{clone}(a)(b)$ in $D$:

$$\Pi_{\{a,b\}}(\rho) = \left( \sum_{v \in V} \langle a, v \rangle \cdot \langle b, v \rangle \cdot \langle b, v \rangle^* \right)^*$$
Accepting execution

An execution $\rho$ is accepted by a data net $D = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$ if it satisfies all the following properties:

Property 6

for each filter element $\text{filt}(a)(b)$ in $D$:

$$\Pi_{\{a,b\}}(\rho) = \left( \sum_{v \in V} \langle a, v \rangle \cdot (\langle b, v \rangle + \epsilon) \right)^*$$
Semantics

Accepting execution

An execution $\rho$ is accepted by a data net $D = \langle N^I_{ext}, N^O_{ext}, N_{int}, D, V \rangle$ if it satisfies all the following properties:

Property 7

for each last element $last(a)(b)$ in $D$:

$$\Pi_{\{a,b\}}(\rho) = \left( \sum_{v \in V} \langle a, v \rangle \right) \cdot \left( \sum_{v \in V} \langle a, v \rangle \cdot \langle b, v \rangle \right) + \epsilon$$
BPEL4WS process execution

\textbf{exec}(\mathcal{W})$: set of all possible ordered sequence of input/output messages (or message parts) received and sent by the process \( \mathcal{W} \) from its activation to its termination.
BPEL4WS process execution

\( \text{exec}(W) \): set of all possible ordered sequence of input/output messages (or message parts) received and sent by the process \( W \) from its activation to its termination.

Data Net Satisfiability

We say that \( W \) satisfies \( D \) when for each process execution \( \rho_W \in \text{exec}(W) \) there exists an accepting execution \( \rho \) of \( D \) such that 
\[
\Pi_{\text{Next}}(\rho) = \rho_W.
\]
Outline

1. Introduction
   - Aim

2. Data Flow Language
   - Syntax
   - VTA Case Study
   - Semantics

3. Automated Composition Framework
Specifying Data-Flow Requirements for the Automated Composition of Web Services

Annapaola Marconi marconi@itc.it

Outline
Introduction
Data Flow Language
Automated Composition Framework

Existing Composition Framework

Abstract

Processes

Requirements

Composition

BPEL4WS

STS2BPEL

EaGLe

SYNTHESYS

BPEL2STS

TRANSLATOR

Component Services

State Transition Systems

W

Composite Service

Wn

W1

BPEL4WS

Abstract Processes

such that

\[ \Sigma_c > \Sigma_{\text{II}} \models \rho \]
New Composition Framework

Specifying Data-Flow Requirements for the Automated Composition of Web Services

Annapaola Marconi marconi@itc.it
Data Requirements as STS

We assume that, in the new composite process, there exists a **variable** for each connection node in \( D \):
We assume that, in the new composite process, there exists a variable for each connection node in $\mathcal{D}$:

- variables associated to external connection nodes are those used by the new composite process to store received messages and to prepare the messages to be sent

- variables associated to internal connection nodes are those used to manipulate messages by means of internal functions and assignments

$\Sigma_{\mathcal{D}}$ defines constraints on the possible operations that the composite process can perform on these variables.
We assume that, in the new composite process, there exists a variable for each connection node in $\mathcal{D}$:

- variables associated to **external connection nodes** are those used by the new composite process to store received messages and to prepare the messages to be sent
- variables associated to **internal connection nodes** are those used to manipulate messages by means of internal functions and assignments
We assume that, in the new composite process, there exists a variable for each connection node in $\mathcal{D}$:

- variables associated to external connection nodes are those used by the new composite process to store received messages and to prepare the messages to be sent
- variables associated to internal connection nodes are those used to manipulate messages by means of internal functions and assignments

$\Sigma_D$ defines constraints on the possible operations that the composite process can perform on these variables.
For each **output operation** of a component service in the \( D \) we define a STS which represents the sending of the message (as an output action) and the storing of all message parts (as internal actions).
For each output operation of a component service in the $D$ we define a STS which represents the sending of the message (as an output action) and the storing of all message parts (as internal actions).

Example

For the output operation $\text{C.request}$ with message parts $\text{date}$ and $\text{loc}$ we define the following STS:
For each input operation of a component service in the $D$ we define a STS which represents the storing of all message parts (as internal actions) and the reception of the message (as an input action).
For each input operation of a component service in the $D$ we define a STS which represents the storing of all message parts (as internal actions) and the reception of the message (as an input action).

**Example**

For the input operation $C$.booked with message parts info and cost we define the following STS:
Data Requirements as STS

We define a STS for each data-flow element in $\mathcal{D}$:
We define a STS for each data-flow element in $D$:

$$\text{id}(a)(b)$$
We define a STS for each data-flow element in $\mathcal{D}$:

**id(a)(b)**

```
  a->b  x->a
```

**oper[f](a,b)(c)**

```
x->b  x->a  f(a,b)->c
x->a  x->b
```
We define a STS for each data-flow element in $D$:

fork($a$)($b,c$)

```
x->a
a->c
a->b
```

merge($a,b$)($c$)

```
x->b
x->a
b->c
a->c
```
We define a STS for each data-flow element in $\mathcal{D}$:
We define a STS for each data-flow element in $\mathcal{D}$:

```
clone(a)(b)
```

Diagram:

- $a \rightarrow b$
- $x \rightarrow a$
- $a \rightarrow b$

The STS $\Sigma^D$, modeling $\mathcal{D}$, is the synchronized product of all the STSs corresponding to external connection nodes and data-flow elements of $\mathcal{D}$.
We define a STS for each data-flow element in $D$:

**clone(a)(b)**

- $a \rightarrow b$
- $x \rightarrow a$
- $a \rightarrow b$

**filt(a)(b)**

- $a \rightarrow b$
- $x \rightarrow a$
- $x \rightarrow a$
- $a \rightarrow b$
We define a STS for each data-flow element in $\mathcal{D}$:
We define a STS for each data-flow element in $\mathcal{D}$:

- **clone(a)(b)**
  
  ```
  a->b  x->a  a->b
  ```

- **filt(a)(b)**
  
  ```
  a->b  x->a  a->b
  ```

- **last(a)(b)**
  
  ```
  a->b  x->a
  ```

The STS $\Sigma_\mathcal{D}$, modeling $\mathcal{D}$, is the synchronized product of all the STSs corresponding to external connection nodes and data-flow elements of $\mathcal{D}$. 
Annapaola Marconi
marconi@itc.it

Specifying Data-Flow Requirements for the Automated Composition of Web Services
... modify STS2BPEL !!!
Evaluation of the Approach

Automated synthesis of the VTA Case Study:

less than 1 sec !!!
Evaluation of the Approach

Automated synthesis of the VTA Case Study:

### CONTROL FLOW REQs
- **TryReach**
  - $c \cdot \text{BOOKED} \land f \cdot \text{BOOKED} \land h \cdot \text{BOOKED} \land m \cdot \text{SUCC}$
- **Fail DoReach**
  - $(c \cdot \text{NOT_AVAL} \lor c \cdot \text{REFUSED} \lor c \cdot \text{HOTEL} \lor c \cdot \text{REFUSED}) \land$
  - $(h \cdot \text{NOT_AVAL} \lor h \cdot \text{CANCELLED} \lor h \cdot \text{START}) \land$
  - $(f \cdot \text{NOT_AVAL} \lor f \cdot \text{CANCELLED}) \land$
  - $m \cdot \text{START}$

### DATA FLOW REQs

Annapaola Marconi
marconi@itc.it

Specifying Data-Flow Requirements for the Automated Composition...
Evaluation of the Approach

Automated synthesis of the VTA Case Study:

... less then 1 sec !!!
The end

Questions?