ITU - Telecommunication Standardization Sector

Delayed Contribution ...
Original: English

STUDY GROUP 10

Lutterworth, 14 - 16 October 1997

Question: Q9/10

SOURCE
France Telecom

TITLE: Structural Concepts in MSC: Architectural description

CONTACT
Régine DJIAN
FRANCE TELECOM - CNET
38-40 Rue du Général Leclerc
92794 ISSY- MOULINEAUX CEDEX 9 - FRANCE
Tél: + 33 01 45 29 54 94
Fax: + 33 01 45 29 40 36
E-mail: regine.djian@cnet.francetelecom.fr

ABSTRACT

This contribution adresses the problem of architecture description in MSC already discussed in a contribution presented by FT/CNET at the St Petersburg meeting (April 1995).

1. General

During St Petersburg meeting, the Group has agreed that it could be very interesting to have a way to represent general knowledge of a set of Msc diagrams.

In the contribution discussed at this meeting, we proposed to include in a MSC document a diagram to represent the architecture of system being described. As MSC Diagrams are closely linked with a SDL specification, we proposed to represent this general knowledge as follows:

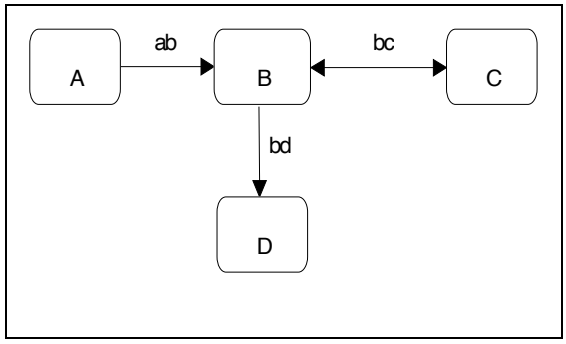


Figure 1: General knowledge representation

A, B, C and D are processes in SDL meaning and instances in MSC meaning.

The figure above shows a general representation of the MSC diagrams below:

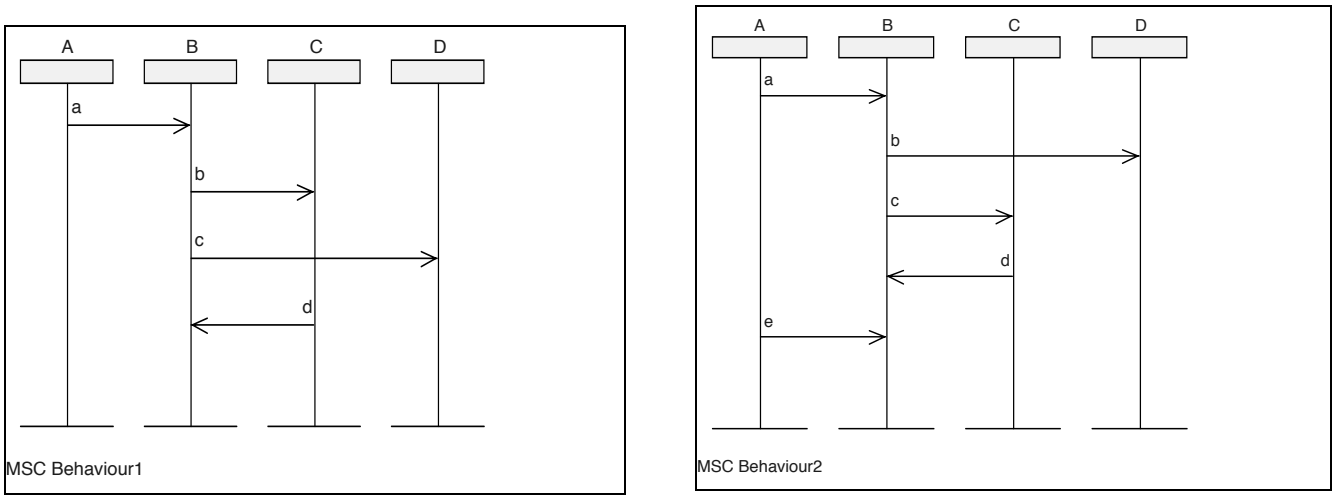


Figure 2: Example of MSC diagrams

In the three following sections we mention first the need of this representation of general knowledge and then advantages for users and tools.

1.1. Purpose

At present MSC are closely used with SDL specification. They represent traces of simulation execution. But we can use MSC before to write a SDL specification. In this case, MSC diagrams are used to describe general behaviour of system and sometimes any error cases. Generally, these diagrams are used to write SDL specifications, but sometimes MSC diagrams are sufficient.

MSC can be used to produce execution traces from SDL simulation or to specify some behaviour for a SDL specification verification, but it can be used to specify test cases by itself. These test cases specifications are intended to be executed on target simulators.

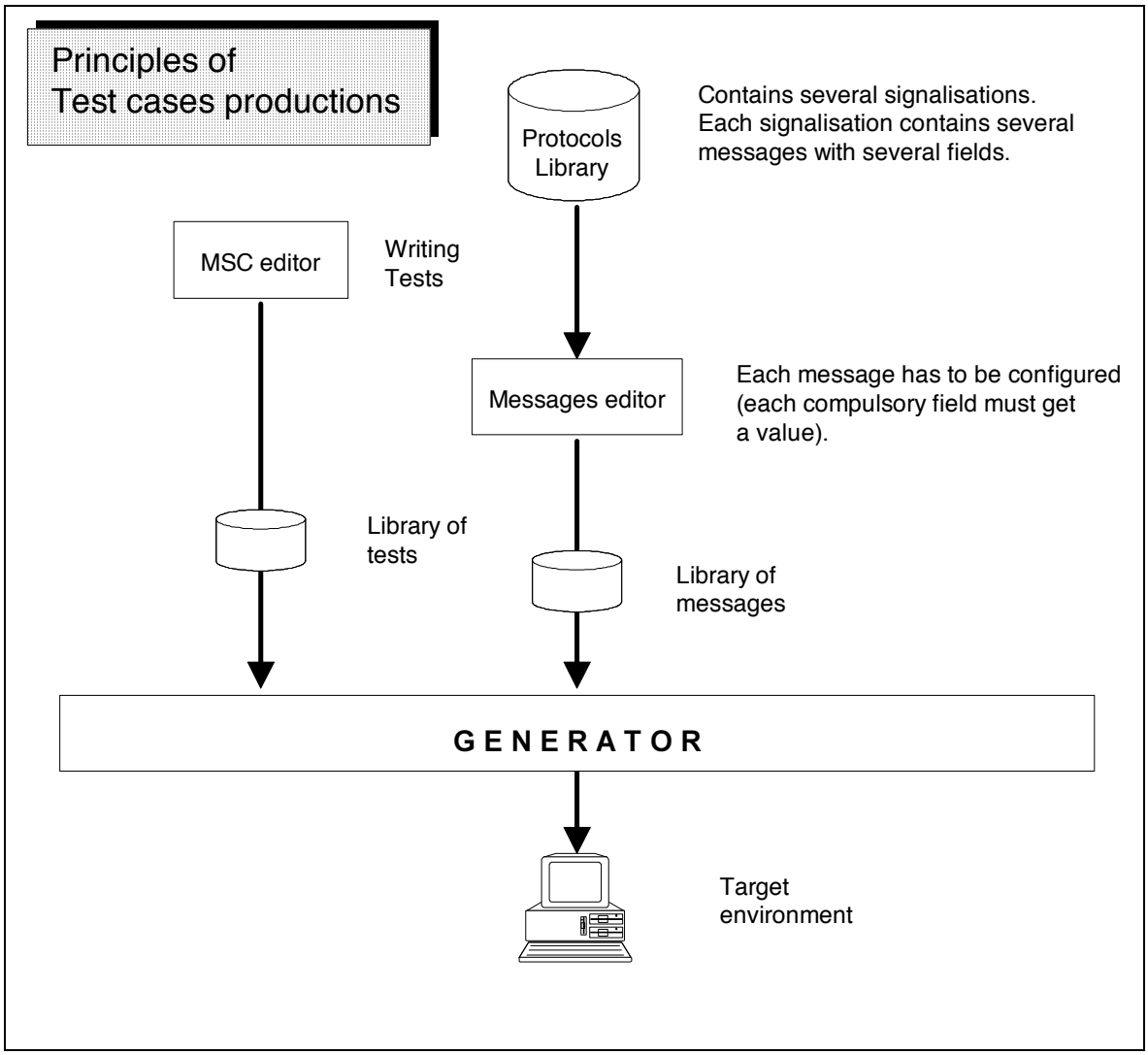


Figure 3: Test Cases Production Environment.

Except graphic aspect allowing an immediate understanding and facilitate the writing of tests, the main advantage to use MSC as validation formalism is that one test can be used to be executed on several systems.

An other advantage is the possibility of reusability of MSC diagrams coming from specification environment (example: SDT or GEODE). So here, MSC specifications must be self-contained. It is why we have to make MSC and SDL independent.

The aim of Architecture is to describe entities of a test and between them flow of information being conveyed. This sub-diagram can be seen as a model of MSC document. User describe this environment only once and for all.

1.2. Advantages for user

The user can obtain correct MSC diagrams with this kind of capabilities. Tools may propose him help to build msc diagrams.
In fact, he builds the architecture of the system he wishes to specify the behaviour only once. This diagram can be used as model to produce several behaviour of same entities.

They can produce more test cases with minimal effort thus saving time and increasing productivity.

1.3. Impact on tools

Tools could take advantage of this architectural description in the following way:

- they may propose help in building a MSC diagram. To each protocol defined as a *<signal name>* may be associated a list of predefined messages. So when the user wishes to create a message between instances A and B, the tool can propose the list of messages allowed to be conveyed between these two instances.

- they may do some semantic checking. First, they can verify if the name of the message that the user specifies is correct or not for the protocol defined between the two instances. Moreover they can take advantage of the direction of the connection defined in the architectural description: they can check if the direction of the message is right or wrong.

Thus user can't produce wrong msc diagram during the building step. Tools make controls during the building step and not after the building step .

In the section below, we propose three ways to introduce this concept in the formalism MSC.
The first one consists in including a subset or the complete part of the SDL description.
The second one consists in including in MSC syntax a little part of SDL syntax.
The three one consists in representing this model with a MSC diagram.
The fourth one consists in a new construct based on SDL BLOCK definition and taking account the problem of decomposed instances.
Whichever the solution we decide to choice, we'll show that MSCs could be improved by a slight modification in the grammar description.

2. Proposals

2.1. Reference to a SDL description

The use of MSC with SDL has become usual practice, but we can imagine to use a MSC document independently of a SDL specification.
So we need a way to make reference to a SDL specification.

At present time, in MSC grammar we have:

related to <sdl reference>

to do reference to a SDL declaration.

If users read a MSC document and only the MSC document they don't know anything about the SDL file this reference comes from.

So we propose to add a new keyword:

import <sdl specification file> <end>

As the <SDL reference>, this rule could be mandatory.

In the following section, the impact on MSC grammar will be exposed.

2.1.1. Textual representation

Concrete grammar on page 22 have to be changed as follow:

<msc document>	::=	<document head> [<sdl import>] <document body>
<sdl import>	::=	**import** <sdl specification name> <end>

2.1.2. Graphical representation

The concrete graphical grammar doesn't need any modification.

2.1.3. Example

In this section, we propose an example to show the manipulation of this representation.

Textual representation:

- In sdlspec.sdl, we have the following block definition.

```
block B;
signalroute S1
   from P1 to P2 with p1p2;
signalroute S2
   from P2 to P3 with p1p3;
   from P3 to P2 with p1p3;
signalroute S3
   from P2 to P4 with p2p4;
process P1 referenced;
process P2 referenced;
process P3 referenced;
process P4 referenced;
endblock B;
```

- A MSC document can refer to this specification as follows:

```
mscdocument mydoc related to system A / block B;
import "sdlspec.sdl";

msc example1;
instance P1;
out a1 to P2;
endinstance;

instance P2;
in a1 from P1;
out b1 to P3;
out b2 to P4;
in c1 from P3;
endinstance;

instance P3;
in b1 from P2;
out c1 to P2;
endinstance;

instance P4;
in b2 from P2;
endinstance;
endmsc;


msc example2;
instance P1;
```

```
out a1 to P2;
out a2 to P2;
endinstance;

instance P2;
in a1 from P1;
out b1 to P4;
out b2 to P3;
in c1 from P3;
in a2 from P1;
endinstance;
endmscdocument;
```

```
instance P3;
in b1 from P2;
out c1 to P2;
endinstance;

instance P4;
in b2 from P2;
endinstance;
endmsc;
```

Graphical representation

The block definition, references the following graphic:

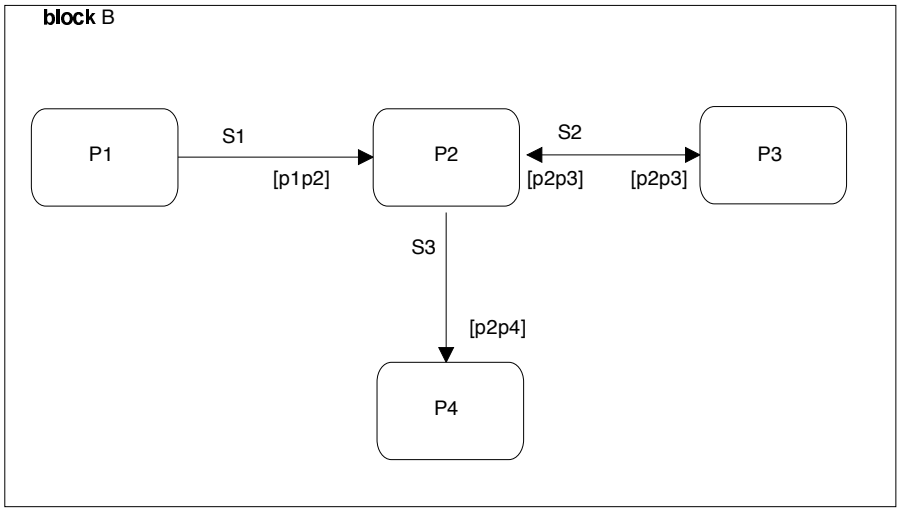
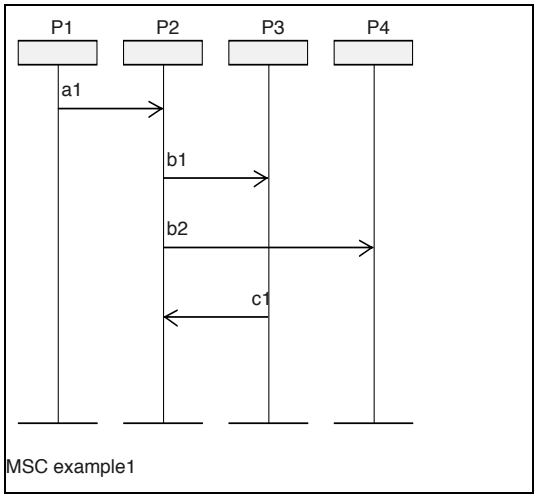


Figure 4: Description of the architecture of the document

And the MSC diagrams corresponding to the textual representation are:



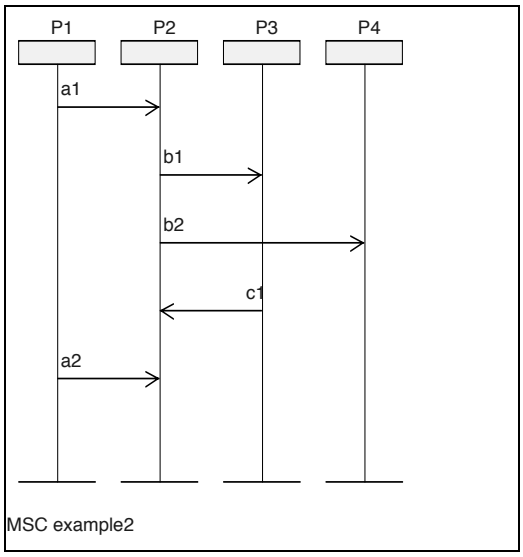


Figure 5: Examples

2.1.4. Comments

The only advantage of this solution is that we only have to modify the textual grammar: we add the possibility to include a special definition of **BLOCK**. This BLOCK is defined in the file specify by <sdl specification name> introduced by the keyword "**related to ...**".

Nevertheless, the main problem that we can see is that a MSC document can't be use alone by a MSC editor. A diagram is nearly linked with a SDL specification and then we need the file containing this specification to use MSC specification.

Then MSC diagram loses its independence. In fact, it's possible to use MSC diagrams in an other way that to produce execution traces of SDL simulation or to specify behaviours to verify SDL specification. MSC diagrams can be used to test system telecommunications to be executed on target machines.

To write tests with MSC allows to reduce the number of tests to write. The main advantage except graphical aspect of MSC allowing to understand immediately the test purpose and to facilitate the writing of the test, is that this test can be used for several systems.

In this case we don't need SDL specification. In the other hand it's quite possible that SDL specification doesn't exist. It is why we propose the solution described in the section bellow.

2.2. Include BLOCK definition in MSC diagram.

Here we use the BLOCK definition of SDL specification. In this case, we can introduce this in a new section: **Document definition section**.
Syntax may be a subset of SDL (Z.100) syntax of BLOCK definition.

2.2.1. Proposal of modifications of Z.120 text

First, concrete textual grammar on page 22 have to be changed as follow:

<msc document>	::=	<msc document head> [<msc document def>] <msc document body>

A new section has to be added before "***4. Basic MSC***". This new section must contain the definition area.

4. **Definition Area.**

A definition area contains the general representation of a set of msc diagrams. Entities and information flows between them are be described. This kind of representation can be seen as a model of document.

This model allows to tools to propose an assistant to build a right diagram.

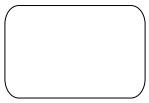
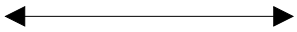
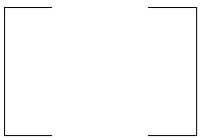
Concrete textual grammar

< msc document def >	::=	**block** {<block name> | <block identifier>} <end> <signal route definition>+ **endblock** <end>
<signal route definition>	::=	**signalroute** <signal route name> <signal route path> [<signal route path>]
<signal route path>	::=	**from** <signal route endpoint> **to** <signal route endpoint> **with** <signal list> <end>
<signal route endpoint>	::=	{<instance name> | **env**} [**via** <gate>]

<signal list>	::=	<signal identifier> {, <signal identifier>}*

After the new text we have proposed above, we have to add :

Concrete graphical grammar

<msc document def area>	::=	<frame symbol> ***contains*** <block def area>
<block def area>	::=	<block heading> <process interaction area>
<block heading>	::=	**block** {<block name> | <block identifier>}
<process interaction area>	::=	{<process area> | <signal route definition area>}+
<process area>	::=	<process symbol> ***contains*** <process name>
<process symbol>	::=	
<signal route definition area>	::=	<signal route symbol> ***is associated with*** {<signal route name> {[<channel identifier> | <external signal route identifier> | <gate>] <signal list area> [signal list area>]} ***set*** ***is connected to*** {<process area> {<process area> | <frame symbol>}} ***set***
<signal route symbol>	::=	<signal route symbol1> | <signal route symbol2>
<signal route symbol1>	::=	
<signal route symbol2>	::=	
<signal list area>	::=	<signal list symbol> ***contains*** <signal list>
<signal list symbol>	::=	
<signal list>	::=	<signal identifier> {, <signal identifier>}*

Semantic

A MSC model is a subset of a SDL BLOCK. A model is mandatory. Only one model is allowed in a document. It describes only instances and connections between them. These connections represent information flow which convey between them. This kind of capability allows to make semantic verifications on msc diagram contained in document.

2.2.2. Example

The example of section 2.1.3 can be adapted in the following way. Only textuel representation changes.

```
mscdocument mydoc;

block B;                                   /* Definition for MSC document of BLOCK */
signalroute S1
   from P1 to P2 with p1p2;
signalroute S2
   from P2 to P3 with p1p3;
   from P3 to P2 with p1p3;
signalroute S3
   from P2 to P4 with p2p4;
process P1 referenced;
process P2 referenced;
process P3 referenced;
process P4 referenced;
endblock B;

msc example1;
instance P1;
out a1 to P2;
endinstance;

instance P2;
in a1 from P1;
out b1 to P3;
out b2 to P4;
in c1 from P3;
endinstance;

instance P3;
in b1 from P2;
out c1 to P2;
endinstance;

instance P4;
in b2 from P2;
endinstance;
endmsc;

msc example2;
instance P1;
out a1 to P2;
out a2 to P2;
endinstance;

instance P2;
in a1 from P1;
out b1 to P4;
out b2 to P3;
in c1 from P3;
in a2 from P1;
endinstance;

instance P3;
in b1 from P2;
out c1 to P2;
endinstance;

instance P4;
in b2 from P2;
endinstance;
endmsc;

endmscdocument;
```

2.2.3. Comments

The main advantage of this solution is to make MSC diagram independent. The graphical solution proposed already exists: it comes from SDL formalism. So tools which use SDL and MSC formalism together can include this new concept easily with a minimal effort.

The solution described in section bellow uses a MSC diagram to describe processes and channels. It is almost based on current MSC standard.

2.3. Definition with a MSC diagram

This third representation consists of a description using a MSC diagram.
To make the distinction between the description of the general knowledge and MSC diagrams, we must introduce a new keyword: **mscmodel** (for example).

2.3.1. Proposal of modifications of Z.120 text

First, concrete textual grammar on page 22 have to be changed as follow:

<msc document body>	::=	[{<msc model> l <msc model diagram>}] {<message sequence chart> l <msc diagram>}*

A new section has to be added before section "***4. Basic MSC***". This new section must contain the model definition.

4. **MSC Model**

A MSC model is a skeleton msc diagram. This diagram contains only instances and messages. Instances represent entities of test and messages represent kind of information which convey between them.

Concrete textual grammar

<msc model>	::=	**mscmodel** <mscmodel name> <end> <msc body> **endmscmodel** <end>
<msc model body>	::=	<msc model statement>*
<msc model statement>	::=	<text definition> l <event definition> l <old instance head statement> <message event>*

Graphical representation is a msc diagram which contains only instances and messages.

Concrete graphical grammar

<msc model area>	::=	<msc symbol> ***contains*** {<msc model heading> <msc model body>}
<msc model heading>	::=	**mscmodel** <mscmodel name>
<msc model body>	::=	{<instance layer> <text layer> <gate def layer> <message area>*}***set***

Semantic

A MSC model is introduced by the keyword **mscmodel**. This new keyword allows to make the difference between the model and MSC diagrams. A model in a MSC document is mandatory. Only one model is allowed in a MSC document. This model describes only instances and connections between them. So the rule <instance event list> will be limited to send and receive messages.
In this case, the name of message represents the type of information which convey between two entities.

2.3.2. Example

The example of sections above can be change as follows:

Textual representation:

mscdocument mydoc **;**

mscmodel mymodel**;**
instance P1**;**
out p1p2 **to** P2**;**
endinstance;
instance P2**;**
in p1p2 **from** P1**;**
outp2p3 **to** P3**;**
in p2p3 **from** P3**;**
out p2p4 **to** P4**;**
endinstance;
instance P3**;**
in p2p3 **from** P2**;**
out p2p3 **to** P2**;**
endinstance;
instance P4**;**
in p2p4 **from** P2**;**
endinstance;
endmscmodel ;

msc example1**;**
instance P1**;**
out a1 **to** P2**;**
endinstance;

instance P2**;**
in a1 **from** P1**;**
out b1 **to** P3**;**
out b2 **to** P4**;**
in c1 **from** P3**;**
endinstance;

instance P3**;**
in b1 **from** P2**;**
out c1 **to** P2**;**
endinstance;

instance P4**;**
in b2 **from** P2**;**
endinstance;
endmsc;

endmscdocument;

msc example2**;**
instance P1**;**
out a1 **to** P2**;**
out a2 **to** P2**;**
endinstance;

instance P2**;**
in a1 **from** P1**;**
out b1 **to** P4**;**
out b2 **to** P3**;**
in c1 **from** P3**;**
in a2 **from** P1**;**
endinstance;

instance P3**;**
in b1 **from** P2**;**
out c1 **to** P2**;**
endinstance;

instance P4**;**
in b2 **from** P2**;**
endinstance;
endmsc;

Graphical representation

The model is described in the following graphic:

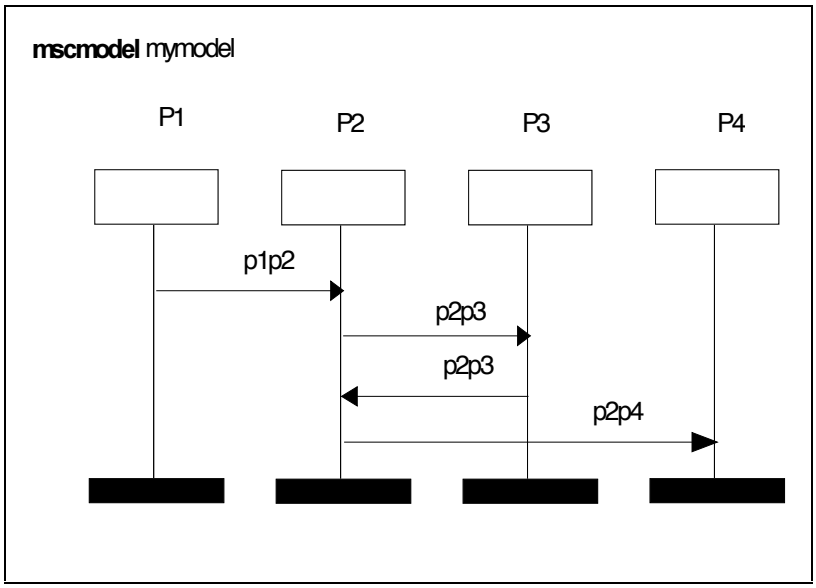


Figure 6: Description of the architecture of the document

Between **P1** and **P2** are allowed messages from protocol **p1p2** and only in the direction P1 to P2.
Between **P2** and **P3** are allowed messages from protocol **p2p3**. The connection is bi-directionnal.
Between **P2** and **P4** are allowed messages from protocol **p2p4** and only in the direction P2 to P4.

And the MSC diagrams corresponding to the textual representation are:

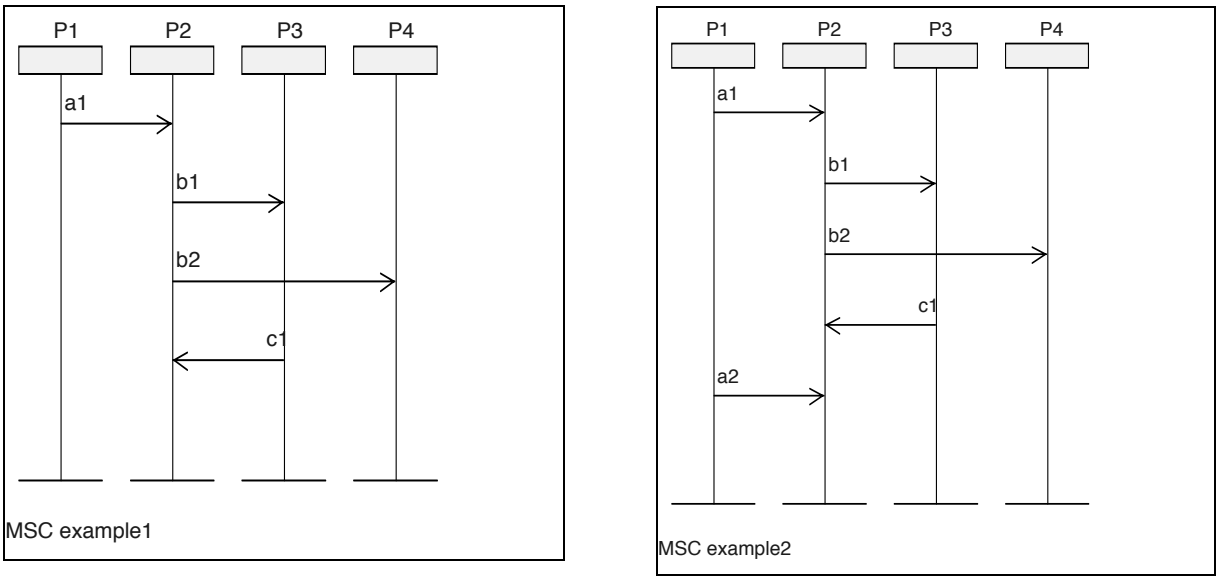


Figure 7: Examples

2.3.3. Comments

The main advantage of this solution is that it doesn't introduce new graphical construct. But this advantage could be a disadvantage. The only element to distinguish between a msc diagram describing the architecture and a msc diagram describing the behaviour of system is the keyword introducing the diagram:
mscmodel ..... **endmscmodel;**

In the other hand, it is not possible to describe all instances of a msc document. We can't represent decomposed instances in an overview diagram.

2.4. A new representation

The main problem of these three proposals is that we can't easily represent the architecture of a decomposed instance. We must represent it in an other diagram. So we lost the main advantage of an architecture diagram which consist of representing whole instances defined in the MSC document and their connections in only one diagram. It is why I propose an other representation which is inspired by the SDL system and block representation. So an architecture can be represented as follow:

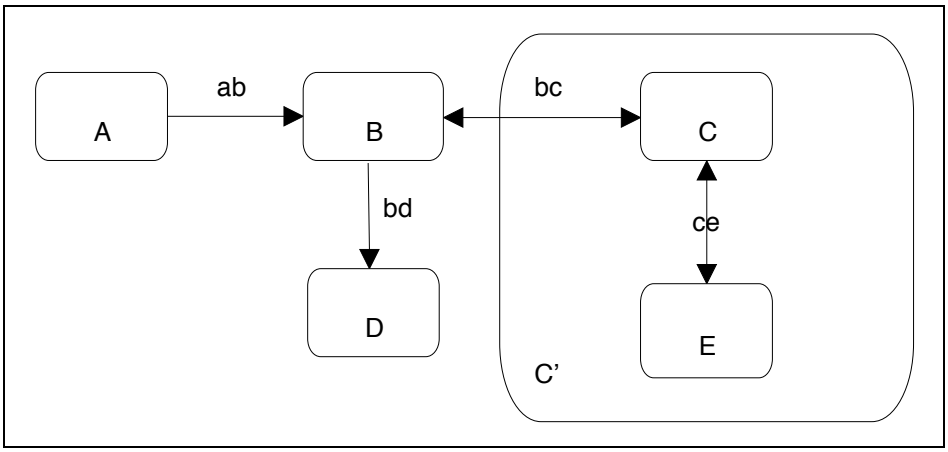


Figure 8: General knowledge with decomposed instance.

Diagram above allows to say that msc diagram contains five instances. And instance C' can be decomposed in instances C and E and messages between B and C' must be members of bc list.

2.4.1. Proposal of modifications of Z.120 text

First, let consider the following definition:

Definition:

A decomposed instance can be considered as a subset of instances defined in architecture diagram. These instances must represent a sub-architecture (cf. *Figure 8*).

In the paragraph below, we propose a modification of Z.120 text.
First, concrete textual grammar on page 22 have to be changed as follow:

<msc document>	::=	<msc document head> [<msc document def>] <msc document body>

A new section has to be added before "***4. Basic MSC***". This new section must contain the definition area.

4. **Definition Area.**

A definition area contains the general representation of a set of msc diagrams. Entities and information flows between them are described. This kind of representation can be seen as a model of document.

This model allows to tools to propose an assistant to build a right diagram.

Concrete textual grammar

< msc document def >	::=	**block** <block name> <end> <signal route definition>+ **endblock** <end>
<signal route definition>	::=	**signalroute** <signal route name> <signal route path> [<signal route path>]
<signal route path>	::=	**from** <signal route endpoint> **to** <signal route endpoint> **with** <signal list> <end>
<signal route endpoint>	::=	{<instance name> [**defined in** <instance name>] | **env**} [**via** <gate>]
<signal list>	::=	<signal identifier> {**,** <signal identifier>}*

After the new text we have proposed above, we have to add :

Concrete graphical grammar

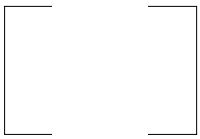
<msc document def area>	::=	<frame symbol> ***contains*** <block def area>
<block def area>	::=	<block heading> <process interaction area>
<block heading>	::=	**block** {<block name> | <block identifier>}
<process interaction area>	::=	{<process area> | <signal route definition area>}+
<process area>	::=	<process symbol> ***contains*** <process name> [<process interaction area>]
<process symbol>	::=	
<signal route definition area>	::=	<signal route symbol> ***is associated with*** {<signal route name> {[<channel identifier> | <external signal route identifier> | <gate>] <signal list area> [signal list area>]} ***set*** ***is connected to*** {<process area> {<process area> | <frame symbol>}} ***set***
<signal route symbol>	::=	<signal route symbol1> | <signal route symbol2>

<signal route symbol1> ::=

<signal route symbol2> ::=

<signal list area> ::= <signal list symbol> ***contains*** <signal list>

<signal list symbol> ::=



<signal list> ::= <signal identifier> {, <signal identifier>}*

Semantic

A MSC model is based on a SDL BLOCK. A model is mandatory. Only one model is allowed in a document. It describes only instances and connections between them. These connections represent information flow which convey between them.

If one instance A is connected to one instance C defined in a decomposed instance C' then messages allowed between A and C' are members of signal list defined between A and C.

This kind of capability allows to make semantic verifications on msc diagram contained in msc document.

2.4.2. Example

In this section, we propose an example to show the manipulation of this representation.

Graphical representation

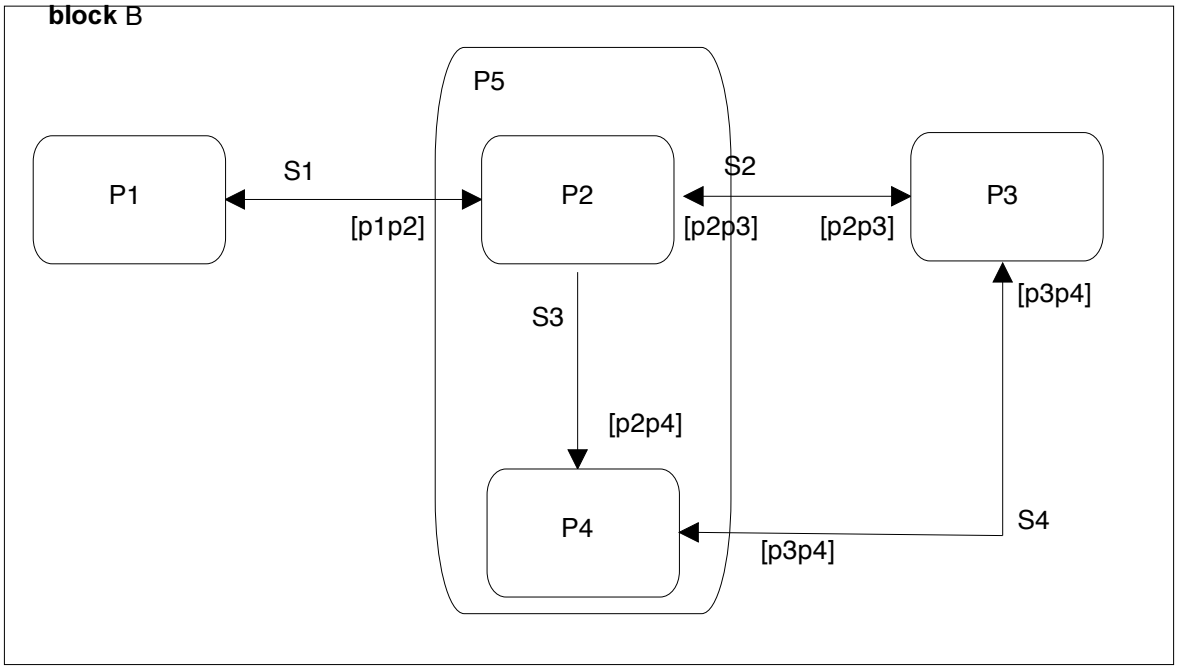
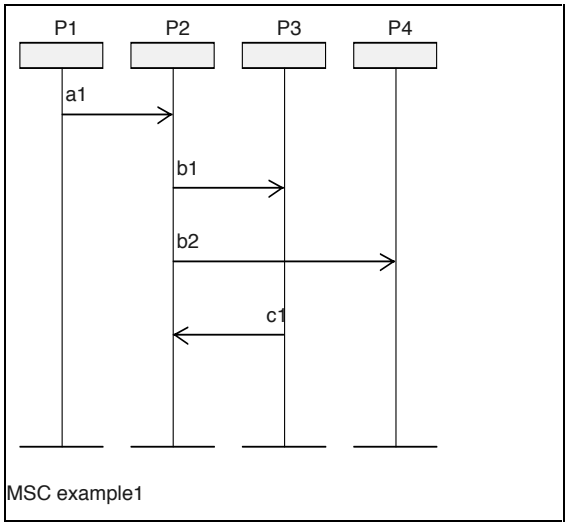
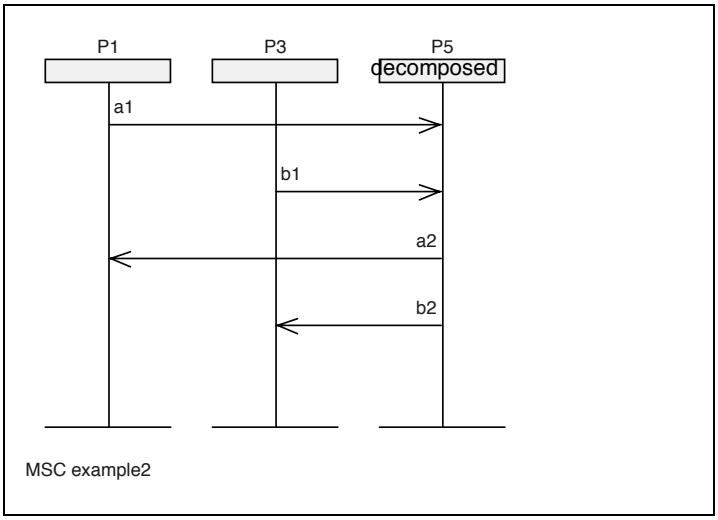


Figure 9: Description of the architecture of the document

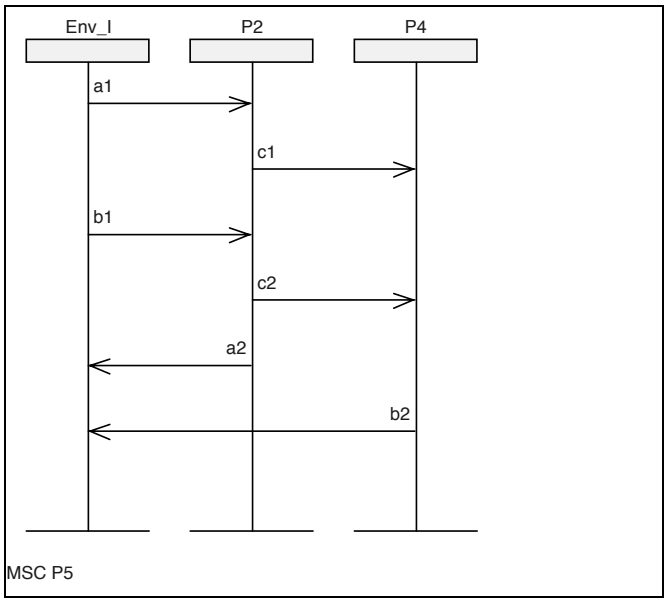
And the MSC diagrams we can build are:





with: a1∈ [p1p2]
b1,c1 ∈ [p2p3]
b2 ∈ [p2p4]

with: a1,a2 ∈ [p1p2]
b1,b2 ∈ [p2p3] ∪ [p3p4]



with: a1,a2 ∈ [p1p2]
b1 ∈ [p2p3]
b2 ∈ [p3p4]
c1,c2 ∈ [p2p4]

Figure 10: Examples

Textual representation

```
mscdocument mydoc;

block B;                                        /* Definition for MSC document of BLOCK */
signalroute S1
   from P1 to P2 defined in P5 with p1p2;
signalroute S2
   from P2 defined in P5 to P3 with p1p3;
   from P3 to P2 defined in P5 with p1p3;
signalroute S3
   from P2 defined in P5 to P4 with p2p4;
endblock;

msc example1;
instance P1;
out a1 to P2;
endinstance;

instance P2;
in a1 from P1;
out b1 to P3;
out b2 to P4;
in c1 from P3;
endinstance;

instance P3;
in b1 from P2;
out c1 to P2;
endinstance;

instance P4;
in b2 from P2;
endinstance;
endmsc;

msc example2;
instance P1;
out a1 to P5;
in a2 from P5;
endinstance;

instance P3;
out b1 to P5;
in b2 from P5;
endinstance;

instance P5 decomposed;
in a1 from P1;
in b1 from P3;
out a2 to P1;
out b2 to P3;
endinstance;
endmsc;

msc P5;
instance P2;
in a1 from env;
out c1 to P4;
in b1 from env;
out c2 to P4;
out a2 to env;
endinstance;

instance P4;
in c1 from P2;
in c2 from P2;
out b2 to env;
endinstance;
endmsc;

endmscdocument;
```

2.4.3. Comments

This solution solves our need and is very simple to understand. The solution already exists because it comes from SDL. So advantages exposed in section 2.2.3 are valid:

- Independence of msc diagram.
- graphical solution exists more or less.
- very easy to implement for existing tools.

3. Conclusion

An architectural description is a simple way to express some properties of the system being described in a MSC document. It could be seen as a general model describing flows of information between entities. Moreover the user could take advantage of this knowledge description during editing. For example, he could obtain a list of messages allowed between two entities or a sub-list of messages only allowed to be sent or only allowed to be received by entities. And tools could use this general information to do some verification of consistency (for example, semantic verification).

The first proposal introduces a notion of reference to an external file. For tools it supposes to read a SDL file. So MSC descriptions and SDL descriptions can't be dissociate. Then we think that this solution is not satisfactory.

The second proposal introduces a new keyword too and a new graphical representation. But these news are not really something new because they come from SDL formalism. So this solution can be integrated by current tools easily. But the main problem is to represent a decomposed instance in the same diagram. SDL doesn't allow this. Nevertheless, I think that we can adapt system and block concepts for our need.

The third proposal introduces a new keyword too. But we don't need to make reference to an external file. And the syntax used to describe this model is the same as the description of a MSC diagram.
The main advantage of this proposal is that a MSC document can be read by user without to know anything about the SDL specification. In an other hand we have to find a new graphical construct to represent channels.

In conclusion, none of these three solutions seems to be satisfactory in the sense that the need is to represent instances and their connections (data type exchanges) in only one diagram. The main problem is to take account decomposed instances. It is why we propose the fourth solution which resolves problems of independence between SDL and MSC specifications and one representation for architectural diagram.