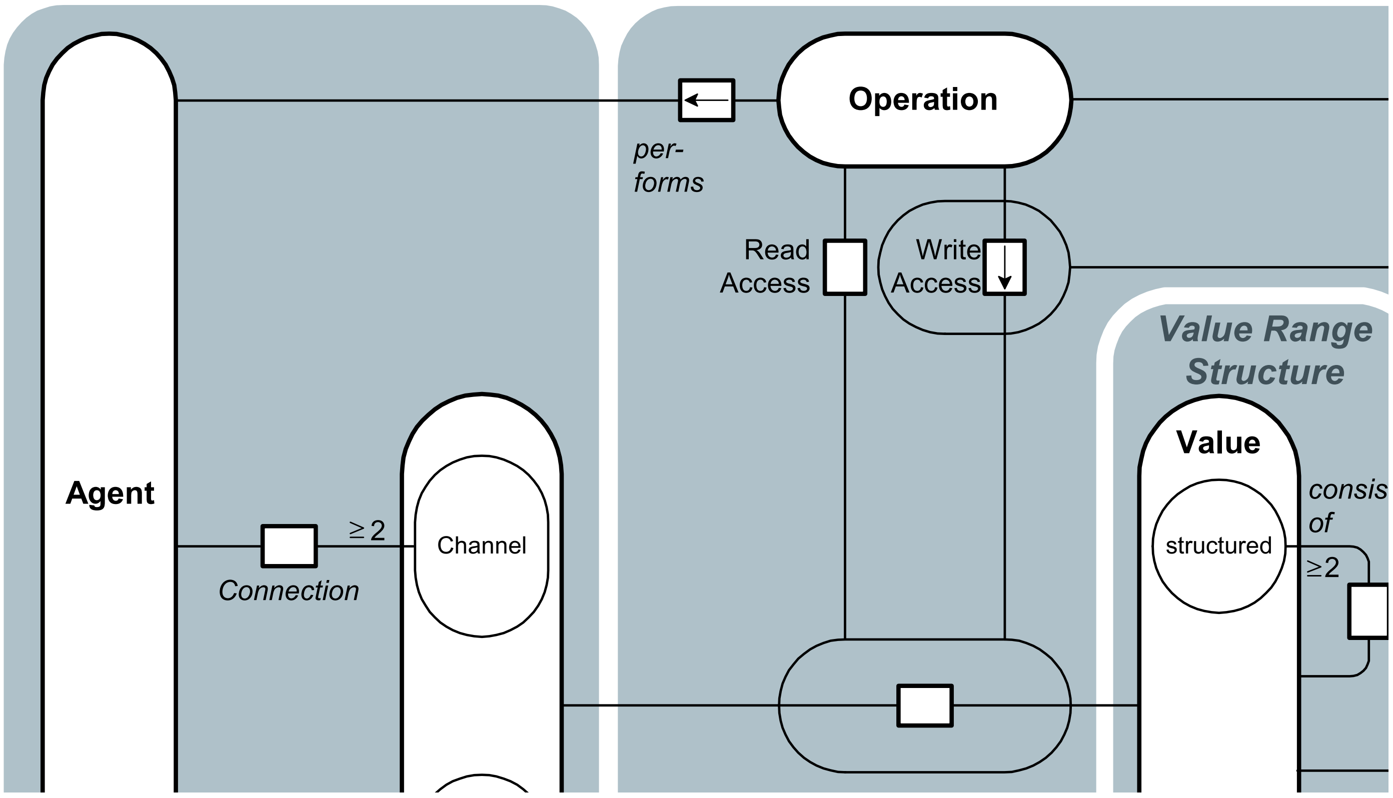
Fundamental Modeling Concepts

http://fmc.hpi.uni-potsdam.de

Peter Tabeling
Rémy Apfelbacher
Stefan Wappler

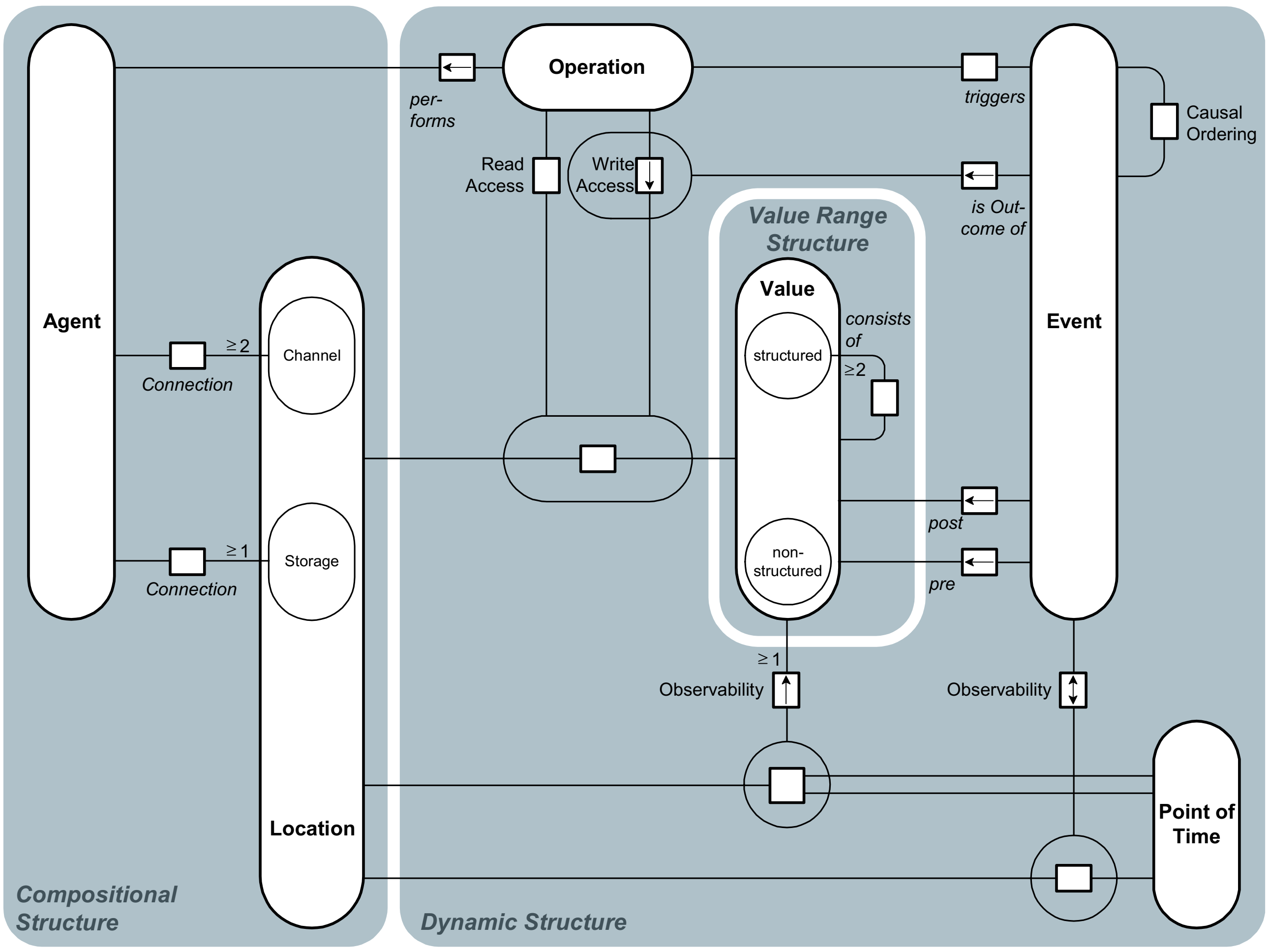
FMC Metamodel

The Fundamental Modeling Concepts Metamodel Explained



Introduction

The FMC metamodel depicts the atomic concepts and their relations which are necessary to describe software-intensive systems using FMC. However, it neither defines the syntax of the different types of FMC diagrams nor the relationships between FMC diagrams on different levels of abstractions.



Comments on the FMC metamodel (as FMC Entity Relationship diagram)

In general, the FMC metamodel distinguishes three fundamental types of structures - **compositional structure, dynamic structure** and **value range structure**. A brief description of these three structures follows below.

Compositional structure

The compositional structure is the (mostly) static structure of a system in terms of components and their connections. A system thus consists of active and passive system components. We call the active components **agents** and the passive ones **locations**. Locations are either **channels** which are only used for communication among the agents, or **storag-**

es which are used by agents to store information. Please note that shared storages (more than one agent having access to a storage) can also be used for communication purposes. The difference between a channel and a storage is that the information on a channel is volatile (e.g. information transmitted by a radio station vanishes as soon as the transmission is stopped) whereas the information in a storage is not (e.g. information on a whiteboard stays until a human agent clears / changes it). A channel is accessed by at least two agents, whereas a storage is accessed by at least one agent. Agents and locations can be both, purely conceptual elements of a high level model or representations of technical components, such as operating system tasks, network connections or storage devices.

Value range structure

While a location is a abstract or physical place, a **value** represents the information that is observable at such a location. A value is either unstructured i.e. is atomic like a bit or an integer number, or structured i.e. it consists of two ore more other unstructured or structured values. Examples for structured values are a record, a stack or even the complete content of a database. Please notice that both, contents of storages and information appearing at channels (messages etc.) are considered as values.

Dynamic structure

The dynamic structure of a system represents all activities performed by agents and their observable results at locations. An **activity** may consist of a single sequence of operations as well as simultaneous sequences of operations. An **operation** is a basic activity, i.e. the "smallest" activity an agent can perform. Normally, when an agent performs an operation it reads values from locations, then it processes this information and finally writes the result to a certain location. Sometimes an agent does not need to read values in order to write values to certain locations, i.e. a generator. So an operation consists of at least one write access concerning at least one location. We may also say that there is at least one access per operation and concerned location. Possible types of **access** to locations are either **read access** which means the value currently observable in a location does not change, or **write access** which means — presupposed the new value differs from the previous one — the value in a location changes. There is also a **modifying access** which consists of a read access followed by a write access to the values of the same location. As an outcome of a write access a value at a location may change, i.e. an **event** occurs. Each event occurs exactly at one **point of time**. The pre- and post-values of the event are the values which can be observed at the location before and after the value change. Accordingly, the post-value of an event can be the pre-value of the following event (at the same location of course). An event itself can cause an agent which is connected to the location where the event occurs to perform an operation. Between events exist causal dependencies, i.e. there is a **causal ordering** (partial ordering).

Relations to FMC diagrams

It is important to point up that the metamodel shows the fundamental terms and their relation. It does not show how systems are described in textual or graphical form with FMC. Nor does it show the syntax of FMC diagrams and the relation between them on different levels of abstraction. The FMC diagrams in general describe types of the three presented structures.
FMC block diagrams indeed depict compositional structures and thus the nodes depict agents and locations. But the edges depict the kind of access (read, write, modifying) which

are not associated to the compositional structure within the metamodel. The FMC Petri nets are used to show the dynamic structures. With the help of Petri nets it is possible to generate occurrence sequences which show the causal ordering of activities / operations / events. There are no graphical representations for access or point of time etc.

Example for the FMC Metamodel

The intention of this section is to introduce an example system in order to give a better understanding of the semantics of the FMC metamodel which is quite hard to understand without an example. The formal description of the example in form of sets and relations is not meant to be really important to understanding. It shows that it is easy to map the system description to the metamodel.

To clarify the metamodel we will present exemplary entity sets and relations resulting of the example system shown in Figure 1. It consists of two agents a1 and a2 which are connected via a channel c and a storage s connected to a2. Agent a1 can send a "count" or a "reset" message via channel c to agent a2. Whenever a message appears at c, agent a2 will increase or reset the value (natural number or zero) currently held at storage s. The value at s is increased modulo 4, i.e. "increasing" 3 yields the value zero. The system thus is a modulo 4 counter with the option to reset the counter.

As one can see, we have defined the entity sets and relationships from the metamodel to reflect the example system of Figure 1. Actually, all entity sets and relations are finite sets which contain only those items that are really present in the system or relevant for the consideration of the system's behavior respectively. (The mathematical descriptions inside the gray shaded boxes may be ignored in order to understand the implications of the example with the metamodel.)

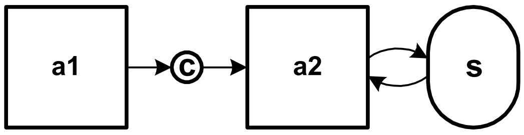


Figure 1: Example system (modulo 4 counter)

Compositional Structure

As we can see our system consists of several system components. The active components – agents – are a1 and a2. A channel c and a storage s are the passive system components. Furthermore agent a1 and agent a2 are connected to channel c, whereas storage s is only connected to agent a2. These are all system components and connections mentioned directly in the description above.

Accordingly, the following sets can be defined:

(Agents) $A = \{a_1, a_2\}$

(Channels) $C = \{c\}$

(Storages) $S = \{s\}$

(Locations) $L = C \cup S = \{c\} \cup \{s\} = \{c, s\}$

(Connections between agents and channels) $C_{AC} \subseteq A \times C = \{(a_1, c), (a_2, c)\}$

(Connections between agents and locations) $C_{AS} \subseteq A \times S = \{(a_2, s)\}$

Value Range Structure

The values appearing at channel c are either "count", "reset" or "empty", where "empty" denotes the value after a message has disappeared and the next one has not yet been sent. The storage s is used to store numbers 0, 1, 2 or 3. Obviously, the values in our example are unstructured.

Thus we can define the following set and relation:

(Values) $V = \{count, reset, empty, 0, 1, 2, 3\}$

(Values placed at locations) $LV \subseteq L \times V = \{(c, count), (c, reset), (c, empty),$
$(s, 0), (s, 1), (s, 2), (s, 3)\}$

Dynamic Structures

Figure 2 shows an exemplary value pattern of the system. It shows information about values, events and points of time on each location but not necessarily the dependencies among them nor which agent is responsible for the occurrence of values. We know from the short introduction of the sample system at the beginning that agent a1 is responsible for changes on channel c. So it has to perform operations to write messages to channel c. Thus agent a1 needs write access to c.

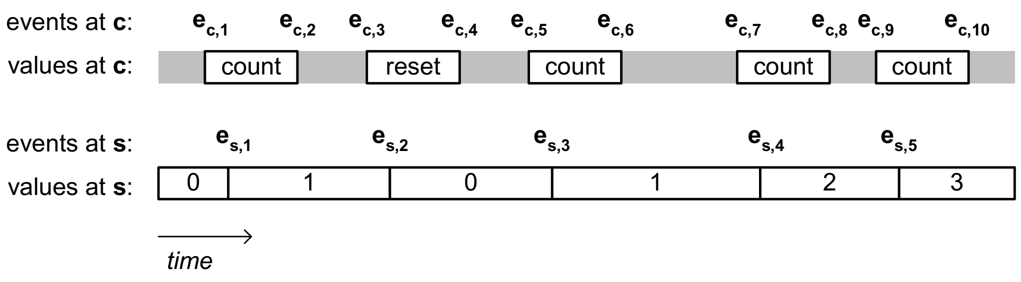


Figure 2: Exemplary system behavior (here value pattern)

Whereas agent a2 performs operations which change the value at storage s depending the message received via channel c. The first operation of a2 is to read the message sent by agent a1, thus a2 needs read access to c. In case of resetting the value at s the operation consists solely of a write access. The operation for increasing ("count") the value at s involves a combined read / write access because a2 needs first to know the current value in order to write back the increased value.

This leads to the definition of following set and relationships:

(Operations)	$O = \{o_1, o_2, o_3, o_4, o_5, o_6, o_7, o_8, o_9, o_{10}, o_{11}, o_{12}, o_{13}, o_{14}, o_{15}\}$
(Operations performed by agents)	$AO \subseteq A \times O = \{(a_1, o_1), (a_2, o_2), (a_1, o_3), (a_2, o_4), (a_1, o_5), (a_2, o_6), (a_1, o_7), (a_2, o_8), (a_1, o_9), (a_2, o_{10}), (a_1, o_{11}), (a_1, o_{12}), (a_1, o_{13}), (a_1, o_{14}), (a_1, o_{15})\}$
(Read access)	$A_{read} \subseteq O \times LV = \{(o_2, (c, count)), (o_2, (s, 0)), (o_4, (c, reset)), (o_6, (c, count)), (o_6, (s, 0)), (o_8, (c, count)), (o_8, (s, 1)), (o_{10}, (c, count)), (o_{10}, (s, 2))\}$
(Write access)	$A_{write} \subseteq O \times LV = \{(o_1, (c, count)), (o_2, (s, 1)), (o_3, (c, reset)), (o_4, (s, 0)), (o_5, (c, count)), (o_6, (s, 1)), (o_7, (c, count)), (o_8, (s, 2)), (o_9, (c, count)), (o_{10}, (s, 3)), (o_{11}, (c, empty)), (o_{12}, (c, empty)), (o_{13}, (c, empty)), (o_{14}, (c, empty)), (o_{15}, (c, empty))\}$

Since an event is defined as the change of a value, an event is the outcome of a write access occurring during one of the operations performed by agent a1 or a2. An event - of course - only occurs if the value written differs from the previous value. Thus a pre and post value is always associated with an event. In our case all write operations change the value of the locations they access. Figure 2 depicts the observable events in our example system.

Some events trigger operations performed by agents. Here, each appearance of a message at channel c triggers an operation (o2, o4, o6, ...) of agent a2 which changes the value at s. These value changes also produce events but these do not trigger any operations because no agent awaits value changes in s.

Events also have a causal ordering specifying which event needs to occur before another one in order for it to occur on its turn.

Thus defining:

(Events) $E = \{ e_{c,1}, e_{c,2}, e_{c,3}, e_{c,4}, e_{c,5},$

$e_{c,6}, e_{c,7}, e_{c,8}, e_{c,9}, e_{c,10},$

$e_{s,1}, e_{s,2}, e_{s,3}, e_{s,4}, e_{s,5}\}$

(Events as outcome of write access) $EA_{write} \subseteq E \times A_{write}$

$= \{ (e_{c,1}, (o_1, (c, count))),$

$(e_{c,2}, (o_{11}, (c, empty))),$

$(e_{c,3}, (o_3, (c, reset))),$

$(e_{c,4}, (o_{12}, (c, empty))),$

$(e_{c,5}, (o_5, (c, count))),$

$(e_{c,6}, (o_{13}, (c, empty))),$

$(e_{c,7}, (o_7, (c, count))),$

$(e_{c,8}, (o_{14}, (c, empty))),$

$(e_{c,9}, (o_9, (c, count))),$

$(e_{c,10}, (o_{15}, (c, empty))),$

$(e_{s,1}, (o_2, (s, 1))),$

$(e_{s,2}, (o_4, (s, 0))),$

$(e_{s,3}, (o_6, (s, 1))),$

$(e_{s,4}, (o_8, (s, 2))),$

$(e_{s,5}, (o_{10}, (s, 3)))\}$

(Pre-values of events) $EV_{pre} \subseteq E \times V$

$= \{ (e_{c,1}, empty), (e_{c,2}, count),$

$(e_{c,3}, empty), (e_{c,4}, reset),$

$(e_{c,5}, empty), (e_{c,6}, count),$

$(e_{c,7}, empty), (e_{c,8}, count),$

$(e_{c,9}, empty), (e_{c,10}, count),$

$(e_{s,1}, 0), (e_{s,2}, 1), (e_{s,3}, 0),$

$(e_{s,4}, 1), (e_{s,5}, 2)\}$

(Post-values of events) $EV_{post} \subseteq E \times V$

$= \{ (e_{c,1}, count), (e_{c,2}, empty),$

$(e_{c,3}, reset), (e_{c,4}, empty),$

$(e_{c,5}, count), (e_{c,6}, empty),$

$(e_{c,7}, count), (e_{c,8}, empty),$

$(e_{c,9}, count), (e_{c,10}, empty),$

$(e_{s,1}, 1), (e_{s,2}, 0),$

$(e_{s,3}, 1), (e_{s,4}, 2), (e_{s,5}, 3)\}$

(Operations triggered by events) $EO \subseteq E \times O$

$= \{ (e_{c,1}, o_2), (e_{c,3}, o_4), (e_{c,5}, o_6), (e_{c,7}, o_8), (e_{c,9}, o_{10}) \}$

(Causal ordering of events) $EE \subseteq E^2$

$= \{ (e_{c,1}, e_{s,1}), (e_{c,3}, e_{s,2}), (e_{c,5}, e_{s,3}), (e_{c,7}, e_{s,4}), (e_{c,9}, e_{s,5}) \}$

The remaining entities and relation all deal with the observability of the system. A hypothetic observer of the system would have access to all locations and would observe the value changes without influencing the system by his observation. The hypothetic observer will only be able to observe:

- Values in locations which are constant in time intervals
- Events in locations at points of time

The definition of these entities and relations would be straight forward but it doesn't give further assistance for the example system. Therefore they are skipped here.