

Towards model-driven communications

Antonio Natali & *Ambra Molesini*

{antonio.natali, ambra.molesini}@unibo.it

ALMA MATER STUDIORUM—Università di Bologna

ICSET 2010, Rome, Italy, 29th April 2010



- 1 The context
- 2 Our Solution
- 3 Conclusions and Future Works



The objective of this paper

- This work intends to discuss how the Model Driven Software Development approach can be considered as a mature technology for



The objective of this paper

- This work intends to discuss how the Model Driven Software Development approach can be considered as a mature technology for
 - ▶ generating in automatic way the schematic part of applications related to communication



The objective of this paper

- This work intends to discuss how the Model Driven Software Development approach can be considered as a mature technology for
 - ▶ generating in automatic way the schematic part of applications related to communication
 - ▶ providing at the same time high level specialized languages useful in all the phases of software production



The objective of this paper

- This work intends to discuss how the Model Driven Software Development approach can be considered as a mature technology for
 - ▶ generating in automatic way the schematic part of applications related to communication
 - ▶ providing at the same time high level specialized languages useful in all the phases of software production
- To achieve the goal



The objective of this paper

- This work intends to discuss how the Model Driven Software Development approach can be considered as a mature technology for
 - ▶ generating in automatic way the schematic part of applications related to communication
 - ▶ providing at the same time high level specialized languages useful in all the phases of software production
- To achieve the goal
 - ▶ a stack of (meta-)languages (meta-meta-models) has been introduced for describing the collaborative behavior of generic entities in terms of communication actions related to a taxonomy of messages



The objective of this paper

- This work intends to discuss how the Model Driven Software Development approach can be considered as a mature technology for
 - ▶ generating in automatic way the schematic part of applications related to communication
 - ▶ providing at the same time high level specialized languages useful in all the phases of software production
- To achieve the goal
 - ▶ a stack of (meta-)languages (meta-meta-models) has been introduced for describing the collaborative behavior of generic entities in terms of communication actions related to a taxonomy of messages
 - ▶ the generation of platforms for communication is viewed as a form of specification of language semantics that provides executable models of applications



Outline

1 The context

2 Our Solution

3 Conclusions and Future Works



Communications

- The analyst and the designer have no reference language to reason about communication (the semantic problem)



Communications

- The analyst and the designer have no reference language to reason about communication (the semantic problem)
- Even if the usage of a specific communication platform is assumed as a working hypothesis (the concept problem)



Communications

- The analyst and the designer have no reference language to reason about communication (the semantic problem)
- Even if the usage of a specific communication platform is assumed as a working hypothesis (the concept problem)
 - ▶ it is uncommon that a platform exposes some formal model of the mechanisms it provides



Communications

- The analyst and the designer have no reference language to reason about communication (the semantic problem)
- Even if the usage of a specific communication platform is assumed as a working hypothesis (the concept problem)
 - ▶ it is uncommon that a platform exposes some formal model of the mechanisms it provides
 - ▶ it is improbable that people spend time for creating a workable model of a platform



Communications

- The analyst and the designer have no reference language to reason about communication (the semantic problem)
- Even if the usage of a specific communication platform is assumed as a working hypothesis (the concept problem)
 - ▶ it is uncommon that a platform exposes some formal model of the mechanisms it provides
 - ▶ it is improbable that people spend time for creating a workable model of a platform
- The semantic gap (the delta problem) between application level and platform level is usually filled up by using OOP to design and build subsystems and components that conceptually belong to the infrastructure rather than to the business level



Communications

- The analyst and the designer have no reference language to reason about communication (the semantic problem)
- Even if the usage of a specific communication platform is assumed as a working hypothesis (the concept problem)
 - ▶ it is uncommon that a platform exposes some formal model of the mechanisms it provides
 - ▶ it is improbable that people spend time for creating a workable model of a platform
- The semantic gap (the delta problem) between application level and platform level is usually filled up by using OOP to design and build subsystems and components that conceptually belong to the infrastructure rather than to the business level
- In absence of a mature software development process, this job is repeated several times, each time in a different way



The semantics problem

- In the Analysis & Design phases each term used to specify a communication pattern must be associated with a *precise semantics*



The semantics problem

- In the Analysis & Design phases each term used to specify a communication pattern must be associated with a *precise semantics*
- That can help not only in finding incoherent system designs, but also in automating (relevant part of) the implementation of the underlying interaction support system



The semantics problem

- In the Analysis & Design phases each term used to specify a communication pattern must be associated with a *precise semantics*
- That can help not only in finding incoherent system designs, but also in automating (relevant part of) the implementation of the underlying interaction support system
- At the best of our knowledge, no language supports analysts and designers in the modeling of the system's interaction dimension at the proper abstraction level

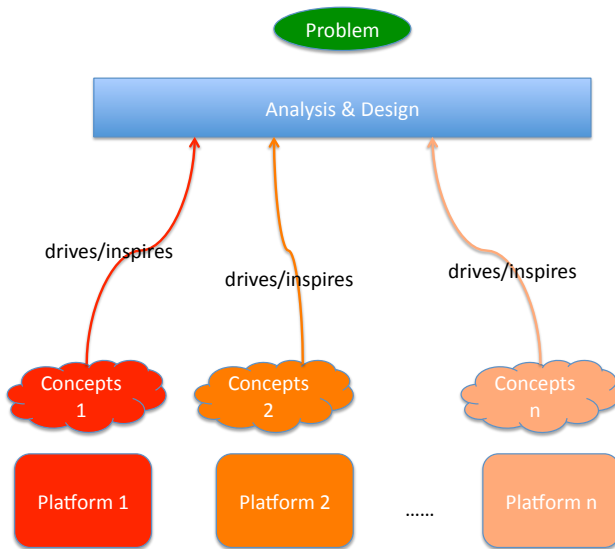


The semantics problem

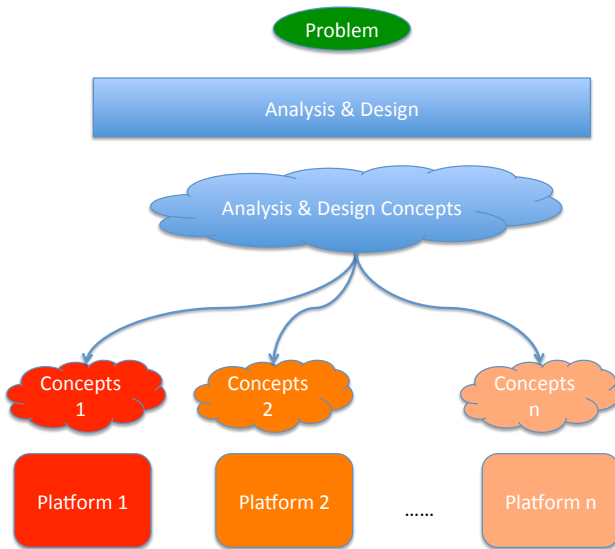
- In the Analysis & Design phases each term used to specify a communication pattern must be associated with a *precise semantics*
- That can help not only in finding incoherent system designs, but also in automating (relevant part of) the implementation of the underlying interaction support system
- At the best of our knowledge, no language supports analysts and designers in the modeling of the system's interaction dimension at the proper abstraction level
- UML 2.0 provides only low-level primitives that are not adapted for modeling the communications at analysis and design level
 - ▶ what is the meaning of UML message?
 - ▶ what is the meaning of synchronous call?
 - ▶ ...



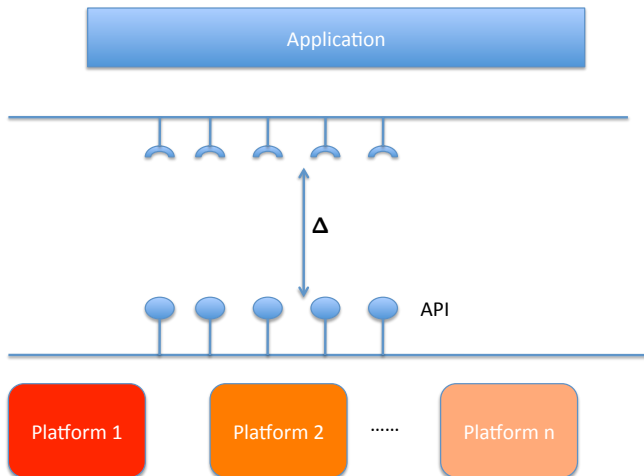
The concept problem



The concept problem



The delta problem



Outline

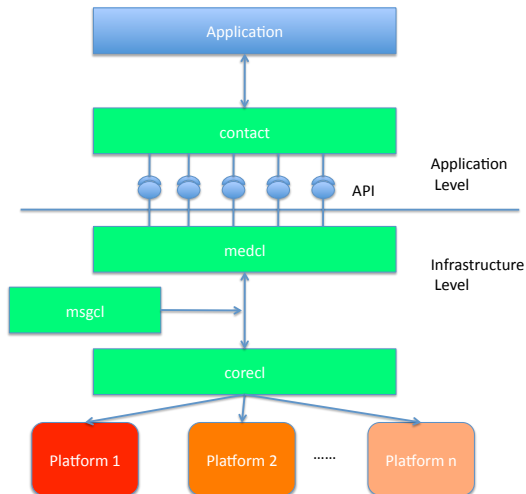
1 The context

2 Our Solution

3 Conclusions and Future Works



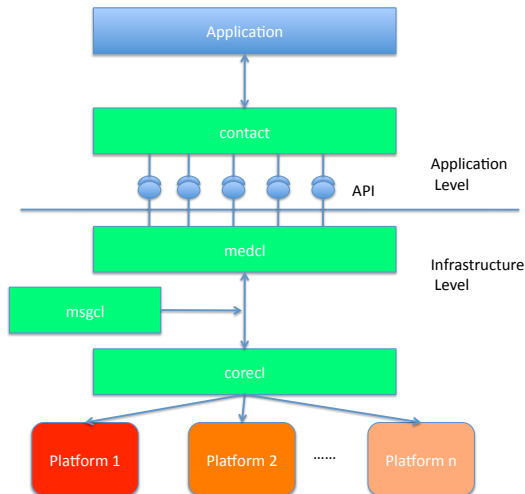
The stack of meta-meta-languages



- **contact**: defines high level language used in the application



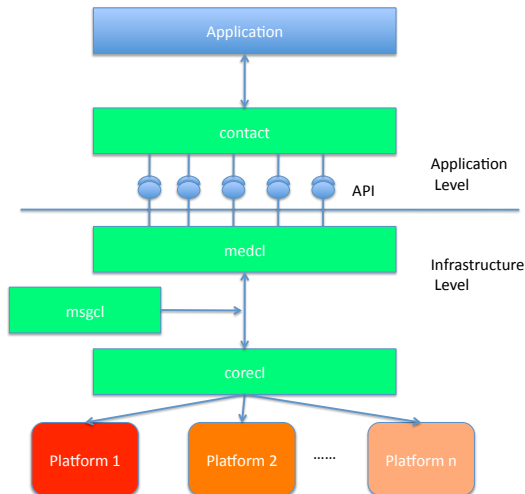
The stack of meta-meta-languages



- **contact**: defines high level language used in the application
- **medcl**: a set of abstract types of communication operations



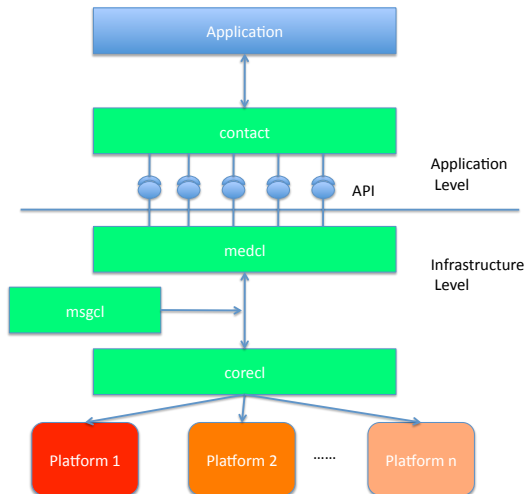
The stack of meta-meta-languages



- **contact**: defines high level language used in the application
- **medcl**: a set of abstract types of communication operations
- **msgcl**: definition of messages and logical channels



The stack of meta-meta-languages



- **contact**: defines high level language used in the application
- **medcl**: a set of abstract types of communication operations
- **msgcl**: definition of messages and logical channels
- **corecl**: abstract low level communication language

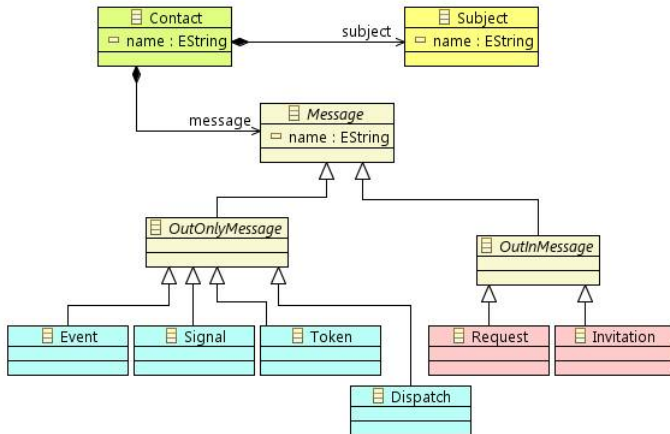


Example

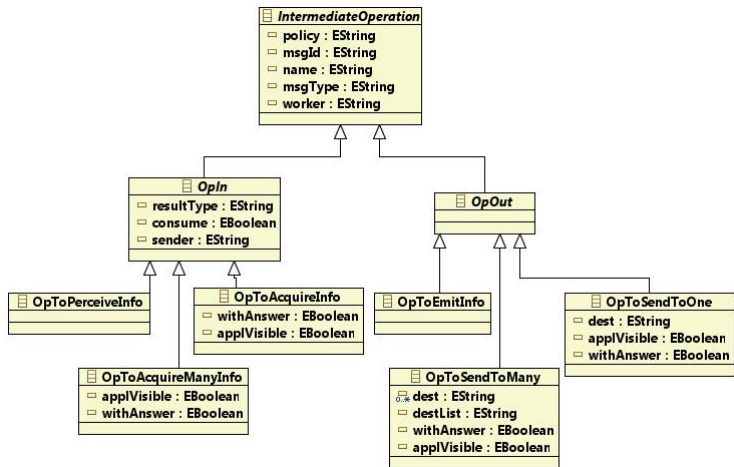
Entity	Instance of	Meaning
User defined program	contact	Language used to specify communication at application level.
MedCI	medcl	Abstract high-level communication language.
Message	msgcl	Language used to define messages and logical channels.
LindaLikeCI	corecl	Abstract low level communication language.



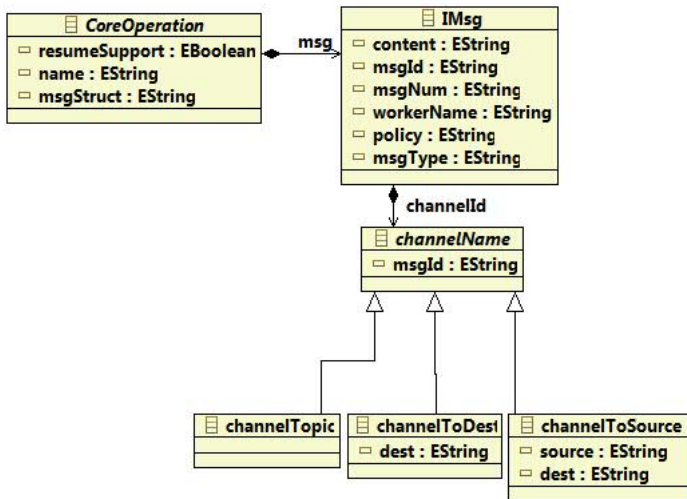
The contact



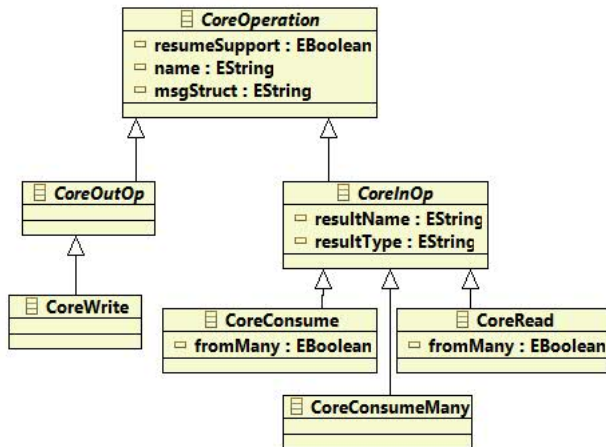
The medcl



The msgcl

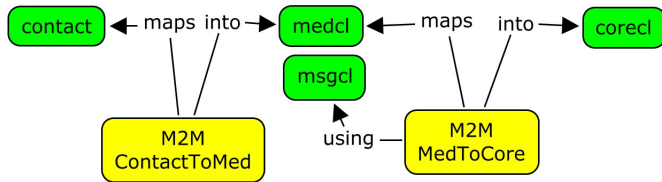


The corecl



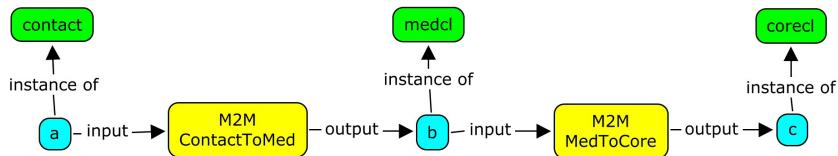
Mapping between the languages

- The first step: mapping a contact action into one `medcl` action via `M2MContactToMed`
- The second step: mapping each `medcl` action into a `msgcl` structure and a sequence of `corecl` actions via `M2MMedToCore`
- The ultimate output of the transformation chain is a model that specifies the behavior of each contact operation as a sequence of `corecl` operations
- This model can be finally translated by a `M2C` generator in the implementation code with reference to some target platform, e.g. Java



Code generation through M2M mapping

- Once defined, the M2M transformers on the meta-meta-models can be used to obtain the semantics of the high-level communication operations written in `contact`



Outline

1 The context

2 Our Solution

3 Conclusions and Future Works



Conclusions

- The first meta-model introduced in this work defines the `contact` language allowing the description of very basic forms of communication
- This language is put at the top of a stack that is currently composed of other two languages, as a consequence, it is possible to generate the schematic part of an application according to a layered logical architecture
- The layer related to the intermediate language (`medcl`) allows to keep the application code independent from the communication medium



Conclusions

- The layer related to the low level language (corec1) keeps the communication medium independent of the implementation platform, by facilitating the mapping of contact on media of different type, e.g. a communication network rather than a shared space
- The three communication issues are resolved:
 - ▶ the semantic problem: the meta-models give a precise semantics
 - ▶ the concept problem: analysis & design have a proper conceptual space independent from the different platforms conceptual spaces
 - ▶ the delta problem: the layered architecture reduce the gap



Future works

- This work is just a starting point to handle high-level communication actions as a sort of *domain-specific language* whose semantics is in some sense “programmable” according to specific application requirements



Future works

- This work is just a starting point to handle high-level communication actions as a sort of *domain-specific language* whose semantics is in some sense “programmable” according to specific application requirements
- `contact`, `msgcl` and `corecl` can be extended to capture more evolute forms of communication
 - ▶ like for example those proposed by FIPA in the agent world
 - ▶ to help in abstracting service-oriented features provided by configurable communication networks



Future works

- This work is just a starting point to handle high-level communication actions as a sort of *domain-specific language* whose semantics is in some sense “programmable” according to specific application requirements
- `contact`, `msgcl` and `corecl` can be extended to capture more evolute forms of communication
 - ▶ like for example those proposed by FIPA in the agent world
 - ▶ to help in abstracting service-oriented features provided by configurable communication networks
- The message ontology discussed in this work could been easily extended so to introduce the concept of `Token` related to very basic communication actions



Future works

- This work is just a starting point to handle high-level communication actions as a sort of *domain-specific language* whose semantics is in some sense “programmable” according to specific application requirements
- `contact`, `msgcl` and `corecl` can be extended to capture more evolute forms of communication
 - ▶ like for example those proposed by FIPA in the agent world
 - ▶ to help in abstracting service-oriented features provided by configurable communication networks
- The message ontology discussed in this work could been easily extended so to introduce the concept of Token related to very basic communication actions
- In this way it is possible to exploit Petri-Nets not only as a formalism to model a software system, but also as a means to concretely build it



Towards model-driven communications

Antonio Natali & *Ambra Molesini*

{antonio.natali, ambra.molesini}@unibo.it

ALMA MATER STUDIORUM—Università di Bologna

ICSET 2010, Rome, Italy, 29th April 2010

