

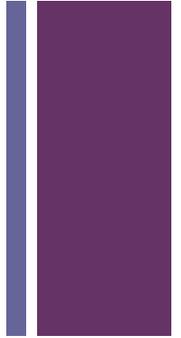
MOMOCS

MDE for the Modernization of Complex Systems

Alessandra Bagnato, **Luciano Baresi**, Francisco Garijo, Jesús Gorroñoigoitia, Matteo Miraz, Juan Pavón Mestras, Giorgio Pezzuto, Luis Quijada, Andrey Sadovykh, Marco Serina, and Jan Vollmar



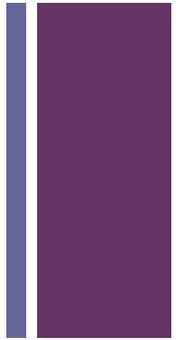
MOMOCS



- Model-driven Modernization of Complex (Software) Systems
- 2-year STREP project (from September 2006)
- Partners
 - TXT eSolutions (Italy)
 - Johann Wolfgang Goethe-Universitat (Germany)
 - Politecnico di Milano (Italy)
 - Softeam (France)
 - Telefonica Investigacion y Desarrollo (Spain)
 - D'Appolonia (Italy)
 - Atos Origin (Spain)
 - Siemens (Germany)
 - SIGS-DATACOM (Germany)



Modernization

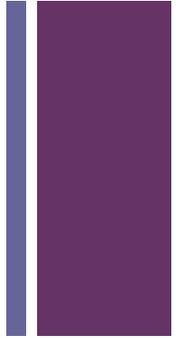


- Many software elements available
 - Important assets for many companies

- Many systems are becoming more and more aged
 - They cannot be substituted completely
 - They are expensive and critical
 - They must be MODERNIZED

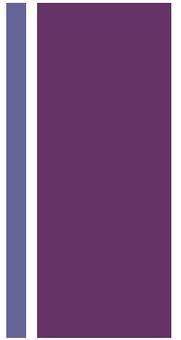
- Many factors
 - Technical, strategic, political, economic ... reasons

+ Is this really new?



- Modernization
 - Maintenance, evolution, porting, ...
 - Now we need a more holistic approach to the problem
 - Important assets must be preserved
- Models and model-driven approaches
 - Around for years
 - Boosted by UML, MOF, and OMG
 - MDA works top-down
 - We move bottom-up

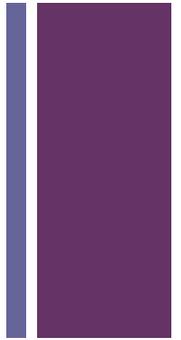
+ XIRUP



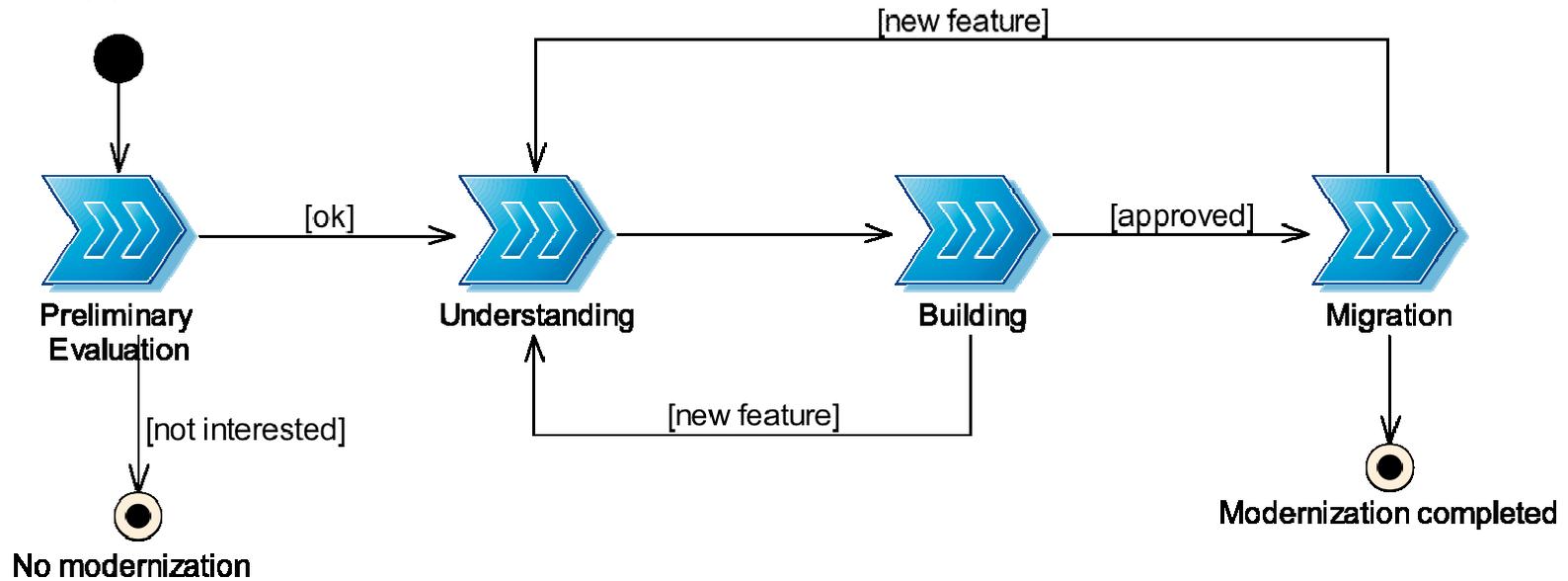
- eXtreme end-User dRiven Process
- Holistic modernization process
 - Along with heuristics and supporting tools
 - From modernization requirements to modernized systems
 - Heavily based on an integrated meta-model
- Model-based approach
 - Existing systems are rendered as models
 - Semi-automatic transformations work on models
 - Models are used to produce the new parts



... in a nutshell



od Xirup process model



+ The scope of KDM (from KDM spec v. 1.0.0)

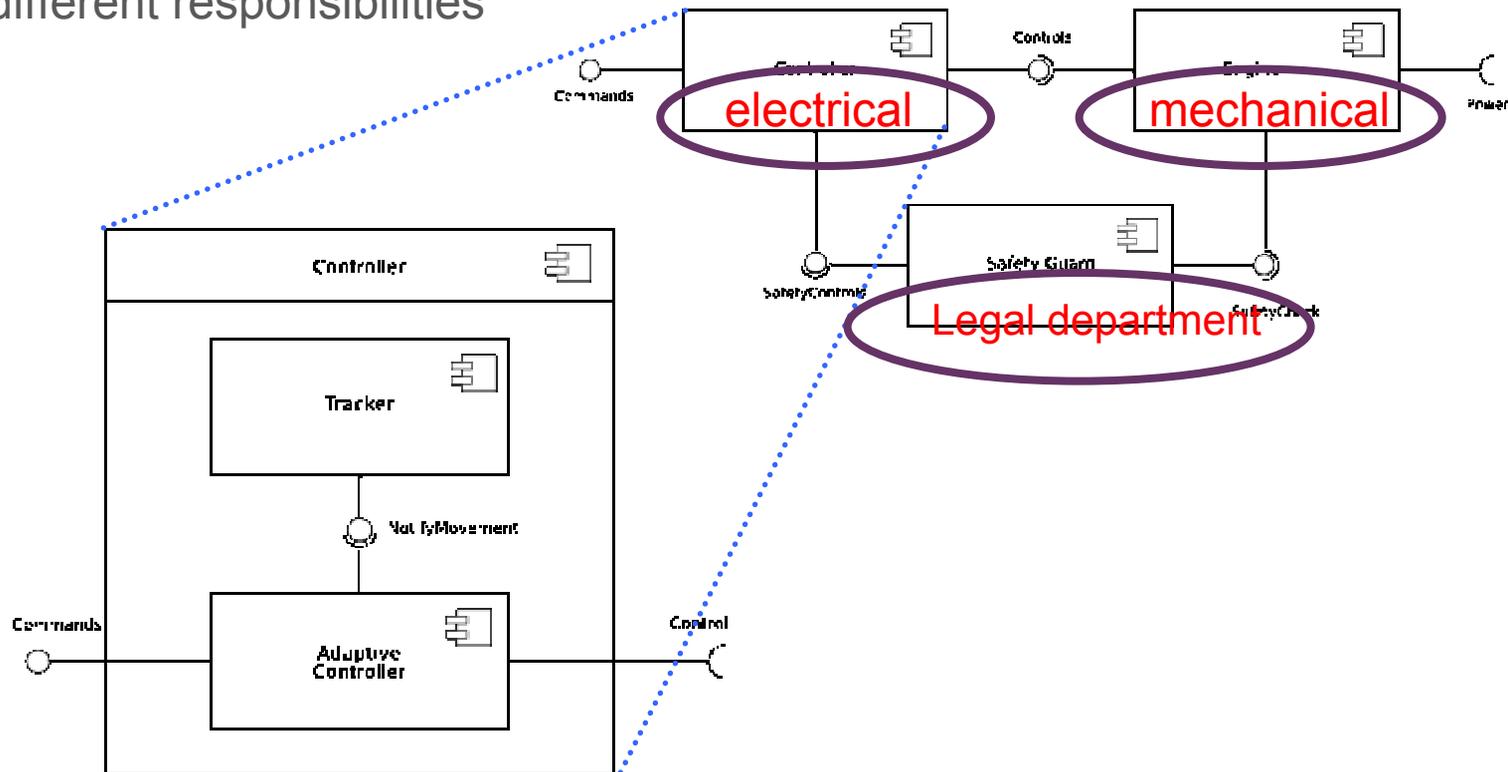
- Defines a meta-model for representing existing software assets, their associations, and operational environments
- Provides a common repository structure that facilitates the exchange of data contained within individual tool models
- Represents the physical and logical assets at various levels of abstraction
- Provides a common interchange format that will allow interoperability between existing modernization and software assurance tools
- Is a MOF model

+ XIRUP meta-model

- Existing meta-models
 - Cope only with software artefacts
 - Do not support agile and component-based modelling
- XIRUP meta-model
 - Describes the collection of things within the domain of modernization of complex systems
 - Supports the evolution and modernization of complex systems
 - Allows different types of engineers to focus on aspects of the system
 - Scales and copes with complex systems
 - Is extensible and adaptable

+ Momocs metamodel

- The complexity is managed:
 - By using an Architectural view on the system
 - At different levels of abstraction
 - With different responsibilities





KB Repository

The screenshot displays the KB Repository tool interface within the Eclipse SDK. The main window shows a UML class diagram for the package `jbilling_ms.umlclass_diagram`. The diagram features a central class `Dynamic.Requires` (with `XSM.Requires` in its name) which is associated with several other classes: `JBillingWrapper`, `Static.Offers`, `Dynamic.Offers`, `Static.Interface`, and `Dynamic.Interface`. At the bottom, three service classes are shown: `ItemSessionHome.Services`, `ListSessionHome.Services`, and `ItemSession.Services`. The interface includes a Navigator on the left showing a tree structure of models, a Properties view at the bottom left, a Palette on the right, and an Ontology view on the far right. The bottom panel shows a transformation diagram with a source model `MOMOC5 D12`, a transformation `<<Manually>>`, and a target model `RM PIM v2`.

+ Modernization patterns

The screenshot displays the Eclipse IDE interface for an ATL project. The main editor window shows the file `ComponentPartReplacement_superimp.atl` with the following code:

```
--This module allows to replace a ComponentPart element with another one having the same type
-- To be used with superimposition
-- uses tags to select
-- To Be Replaced Tag
-- To Be Replaced TagT
-- Replacing Part TagT
-- Replacing Part Tag:

module ComponentPartReplacement
create OUT : Xirup fr

--check if a ComponentPart
helper context Xirup fr
if (self.tag->exists(
then
true
else
false
endif;

--check if a ComponentPart
helper context Xirup fr
if (self.tag->exists(
then
true
else
false
endif;

--get the TBR ComponentPart
helper def : getTBRPart : Xirup!ComponentPart =
Xirup!ComponentPart->allInstances()->iterate(e,res : Sequence(Xirup!ComponentPart) = Sequence{}|
if e.hasTBRTag()
then
```

The `The pattern` dialog box is open, showing a graphical explanation of the pattern. It features two diagrams of `Type AdminPart` activity diagrams. The left diagram shows the original structure with `Part Replacing TravelBean` and `Tag Tag RP` circled in red. The right diagram shows the modified structure with `Part AdminDashBoard` and `Tag Tag RP` circled in red. A red arrow points from the circled `Tag Tag RP` in the left diagram to the circled `Tag Tag RP` in the right diagram.

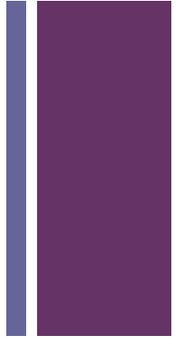
The Navigator on the left shows the project structure, including the `resources` folder and various ATL files. The Outline on the right shows the module's contents, including `OUT : OclModel`, `IN : OclModel`, and several helper and rule definitions.

The ATL operations window at the bottom left shows a list of operations, including `union(c : Collection)`, `flatten()`, `append(o : oclAny)`, `prepend(o : oclAny)`, `insertAt(n : Integer, o : oclAny)`, `subSequence(lower : Integer, upper : Integer)`, `at(n : Integer)`, `indexOf(o : oclAny)`, `first()`, `last()`, `including(o : oclAny)`, and `excluding(o : oclAny)`.

The Problems window at the bottom shows 0 errors, 0 warnings, and 0 infos. The Console window is empty.



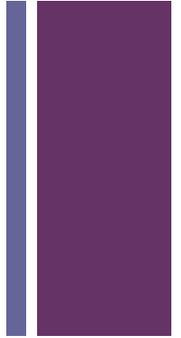
Conclusions and future work



- First release of supporting tools available
 - First feedback on the way
- More integration among the different parts
 - After initial feedback from the consortium
 - Further refinement of supplied features
- Future work
 - Further generalization
 - Automatic import functionality for different technologies
 - Liaisons with OMG



Questions



- MOMOCS project
- Alessandra Bagnato
 - c/o TXT e-solutions
via Frigia, 27 – 20126 Milano, Italy
alessandra.bagnato@txt.it