Integrating Role Activity Diagrams and Hybrid IDEF for Business Process Modeling Using MDA*

Costin Bădică, Maria Teodorescu, Cosmin Spahiu, Amelia Bădică
University of Craiova, Software Engineering Department
Bvd.Decebal 107, Craiova, 200440, Romania
badica_costin@software.ucv.ro

Chris Fox
University of Essex, Computer Science Department
Colchester CO4 3SQ, United Kingdom
foxcj@essex.ac.uk

Abstract

Business process modeling is an important phase during requirements collection. Usually functional, dynamic and role models are needed. We propose to integrate Role Activity Diagrams and Hybrid IDEF for business process modeling within Model Driven Architecture. Our proposal is demonstrated with a sample implementation.

1. Introduction

Various notations are employed for capturing functional, dynamic and role-based aspects of a business process. We propose to integrate two notations for business process modeling – Role Activity Diagrams (RAD, [11]) and Hybrid IDEF ([7]) within the Model Driven Architecture (MDA). Our proposal is demonstrated using a sample implementation based on Eclipse platform ([1]). The added-value of our proposal is to bring the possibility of approaching business process modeling from either a mixed functional/dynamic view using Hybrid IDEF, or from a more human-like/role-based view using RAD and still be able to switch later this view depending on changes in the modeling focus.

2. Overview of RAD and Hybrid IDEF

2.1. Role Activity Diagrams

RAD is a visual notation for business process modeling ([11]). RAD is useful for modeling organized human behavior and interactions ([9]). RAD is also at the basis of RADRunner – a software that supports cooperation, collaboration and communication between people ([2]). Figure 1 shows an example RAD model of a business process for carrying out a design project.

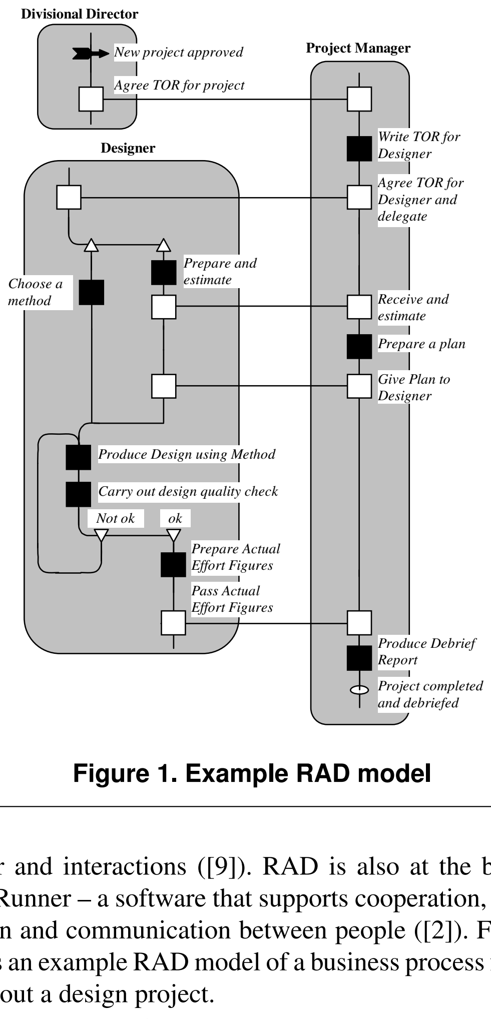


Figure 1. Example RAD model

The elements of a RAD model are: *roles* – group together activities into units of responsibility; *part refinements*

* This work was carried out as part of the CNCSIS grant 94/2005

– describe parallel execution threads; *case refinements* – describe choices; *activities* – basic building blocks for describing work; *interactions* – activities requiring coordination with activities in other roles; *external events* – points at which state changes occurring in the environment influence on our process; *states* – useful to model point wise process goals; *synchronization points* – needed to synchronize threads originating from the same part refinement (see [11, 8] for more details).

2.2. Hybrid IDEF

Hybrid IDEF integrates IDEF0 for function modeling ([5]) and IDEF3 for dynamic modeling ([10]) into a unified and consistent business process modeling notation. IDEF0 and IDEF3 are also supported by modeling and simulation tools – AI0 WIN and ProSim ([3]).

The basic modeling element of Hybrid IDEF is the labeled black box that models an activity with inputs and outputs. A glass box view that models the logic of inputs selection and outputs production by means of input and output trees of connectors is attached to each black box. Resources (humans or machines) are assigned to black boxes via mechanisms. Resources are grouped into pools attached to *bundled mechanisms*. A bundled mechanism is split into a set of mechanisms linked to activities. The splits model resource parts taken from available pools and needed to carry out activities (see [7] for details). Hybrid IDEF allows the description of a business process model at various levels of detail. Here, we assume that a Hybrid IDEF model comprises only a given intermediate level of detail of the target business process and we call this a *diagram*.

An example Hybrid IDEF model that shows only the black-box view is presented in figure 2.

3. Mappings RAD ↔ Hybrid IDEF

Here we give an informal description of mappings between RAD and Hybrid IDEF. Note that in this section, by model we understand a RAD model and by diagram we understand a Hybrid IDEF diagram.

3.1. Mapping RAD to Hybrid IDEF

A model is mapped to a diagram containing all RAD activities. A single default bundled mechanism that binds all diagram mechanisms flows is created. A role is mapped to a mechanism that denotes an active resource and is part of the default bundled mechanism.

An external event is mapped to a diagram input. An activity is mapped to a single output IDEF activity. A case refinement is mapped to an activity. Its output tree consists of an output XOR connector and a set of leaves, one leaf for each condition. A part refinement is mapped to an IDEF activity. Its output tree consists of an output AND connector and a set of leaves, one leaf for each thread. A synchronization point is mapped to a single output IDEF activity. The root of its input tree is an AND input connector. An interaction is mapped to an IDEF activity. The roots of its input and output trees are of type AND and have arity equal to the number of participant roles.

Finally, we must describe how input trees of generated IDEF activities are constructed. Note that this was just partly covered by the above description. A state is mapped to an input tree. The input tree of an activity derived from a RAD activity, part refinement or condition is generated from the state adjacent to that element in the RAD graph. This tree is constructed recursively taking into account preceding external events and states in the RAD graph. We have two cases: i) An external event is interpreted as having the semantics of a diagram input. Therefore, it generates a leaf of an input tree with AND root; ii) A state with *n* children in the RAD graph is mapped to an input tree with XOR root and *n* children that are generated recursively from the nodes adjacent to that state.

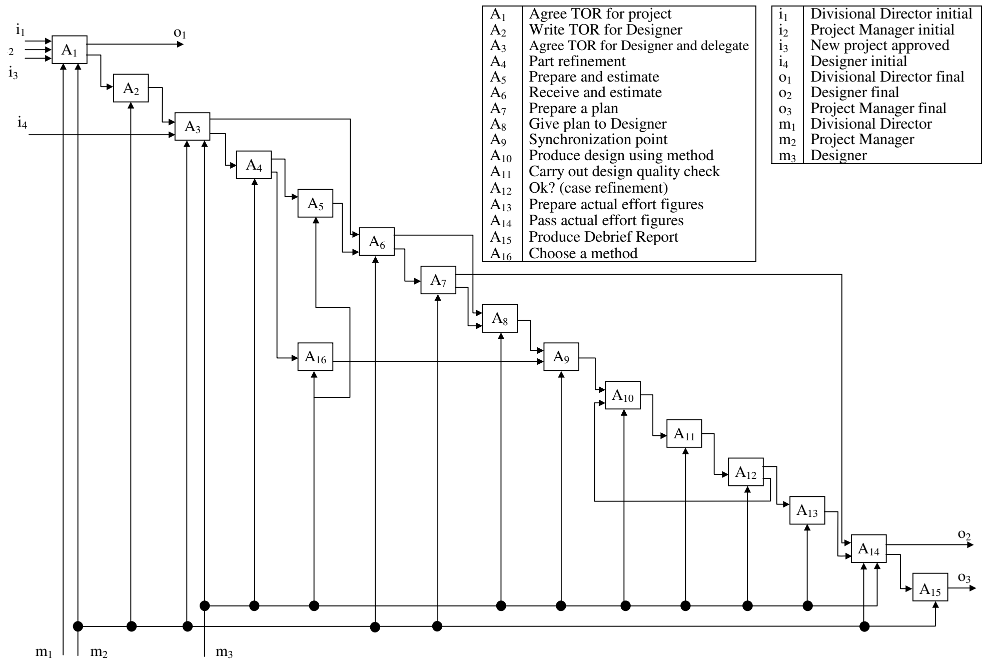
Figure 2 shows the Hybrid IDEF diagram resulted by mapping the RAD model from figure 1.

3.2. Mapping Hybrid IDEF to RAD

A diagram is mapped to a model that contains roles derived from diagram mechanisms. A mechanism is mapped to a role if it denotes an active resource. The role will contain elements resulted from activities that refer the source mechanism and connectors of their input and output trees. An activity is replicated in all roles corresponding to active mechanisms that it refers.

An activity is mapped to a RAD activity if it has a single active mechanism or to an interaction if it has at least two active mechanisms. If the activity was not assigned an active mechanism then it is deleted. Flows are mapped to states and/or external events. For each diagram input an external event is generated. A diagram output is mapped to a final state. Input and output trees are mapped recursively to part refinements, case refinements, synchronization points and states. An input AND connector is mapped to a synchronization point. An input XOR connector is mapped to a state with multiple predecessors. An output AND connector is mapped to a part refinement. An output XOR connector is mapped to a case refinement.

Finally, we must relate nodes of different roles that are resulted by mapping a flow of an IDEF diagram that connects two activities assigned to different mechanisms. For this purpose we use a state for the source activity and an external event for the destination activity with equal names derived from the original flow.



A_1	Agree TOR for project
A_2	Write TOR for Designer
A_3	Agree TOR for Designer and delegate
A_4	Part refinement
A_5	Prepare and estimate
A_6	Receive and estimate
A_7	Prepare a plan
A_8	Give plan to Designer
A_9	Synchronization point
A_{10}	Produce design using method
A_{11}	Carry out design quality check
A_{12}	Ok? (case refinement)
A_{13}	Prepare actual effort figures
A_{14}	Pass actual effort figures
A_{15}	Produce Debrief Report
A_{16}	Choose a method

i_1	Divisional Director initial
i_2	Project Manager initial
i_3	New project approved
i_4	Designer initial
o_1	Divisional Director final
o_2	Designer final
o_3	Project Manager final
m_1	Divisional Director
m_2	Project Manager
m_3	Designer

Figure 2. Hybrid IDEF diagram (black-box view) resulted by mapping from RAD model shown in figure 1

4. System Description

Here we provide a brief description of a prototype system based on Eclipse plug-ins for RAD and Hybrid IDEF model acquisition and transformation.

4.1. RAD and Hybrid IDEF meta-models

MDA is an approach of Object Management Group to software development based on modeling and model mapping ([4]). MDA defines platform-independent models (PIM), platform-specific models (PSM) and mappings between them. We found natural to map business processes of an organization to PIMs. Therefore we have PIMs for RAD and Hybrid IDEF and their consistent integration is achieved by defining corresponding mappings.

Figure 3 shows the RAD meta-model. A *Model* is a set of role models. A *Role* represents a single RAD role as a directed graph that acts as a container of nodes (*Node* class). There are four kind of nodes: *State*, *CaseRefinement*, *Interaction* and *Action*. *Action* nodes are further subdivided into activities, part refinements, external events and synchronization points according to *type* attribute. A *Role* contains the nodes of the role graph (*listNodes* containment association) and points to the start states (*firstNodes* reference association). A *Node* comprises the list *nextNodes* adjacent from it and it points to the referencing roles (*parentRole* association). Cases of a case refinement are linked together via *nextCase* association. Actions and interactions allow the

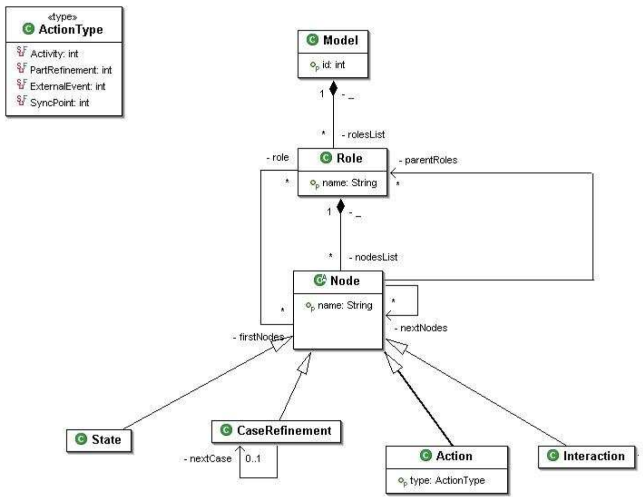


Figure 3. RAD meta-model

specification of passive resources required to carry out underlying work (*resourceDescription* attribute).

Figure 4 shows the Hybrid IDEF meta-model. A *Diagram* contains activities and bundled mechanisms. A *BundledMechanism* is split into several mechanisms (association *mecs*). A mechanism represents a passive or an active resource (*active* attribute). An *Acrtivity* refers its mechanisms and its input and output trees (*mecList*, *inputTree* and *outputTree* associations). Input and output trees con-

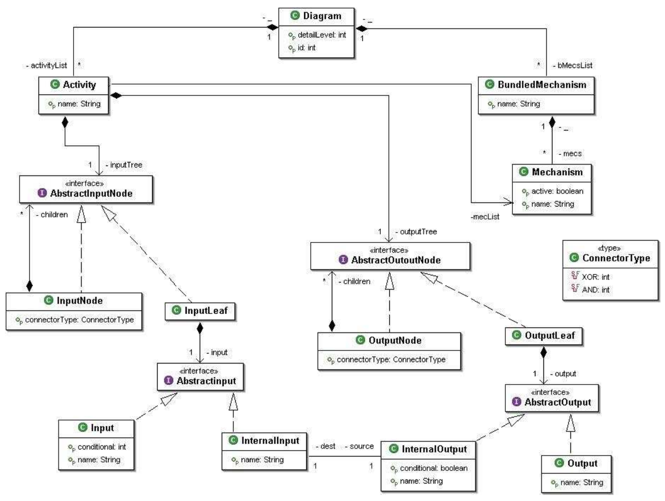


Figure 4. Hybrid IDEF meta-model

tain nodes that are represented by classes *AbstractInputNode* and *AbstractOutputNode*. Tree nodes are classified into internal nodes (classes *InputNode* and *OutputNode*) and leaf nodes (classes *InputLeaf* and *OutputLeaf*). Internal nodes have type AND or XOR (*connectorType* attribute) and are linked to their children via *children* association. Leaf nodes contain abstract input (class *AbstractInput*) and respectively output objects (class *AbstractOutput*) derived from diagram data flows. They are further subdivided into diagram input and output flows (classes *Input* and *Output*) and internal flows (classes *InternalInput* and *InternalOutput*). An internal input is the destination of an internal flow and an internal output is the source of an internal flow (associations *dest*, *source*).

4.2. Model Acquisition Plug-ins

Models are acquired using Eclipse plug-ins that are automatically generated by the Eclipse Modeling Framework (EMF) – a model engineering extension of Eclipse. EMF consists of two fundamental frameworks: *core framework* – provides the basis for creating a Java implementation of a model; *edit framework* – extends and builds on the core framework by adding support for viewing and command-based editing of a model and a basic model editor. EMF is also based on a model called Ecore that is the meta-model for all application models that are handled by EMF.

The process of developing RAD and Hybrid IDEF specialized editors consisted of the following steps: i) Creation of the RAD and Hybrid IDEF meta-models; ii) Description of meta-models as Java interfaces with additional annotations to capture model information that is not expressible directly in Java; iii) Generation of Ecore representations of RAD and Hybrid IDEF models and editor plug-ins. The generated plug-ins provide tree editors of RAD and Hybrid IDEF models.

4.3. Transformation Plug-ins

The transformations were implemented using the ATLAS Transformation Language – ATL ([6]). ATL is a hybrid language that mixes declarative and imperative constructions necessary for expressing model transformations as required by the MDA approach.

A transformation plug-in allows the user to choose the transformation (IDEF to RAD or RAD to IDEF), the source model file and the target model file and launches the ATL engine to perform the transformation. A transformation requires the following: ATL transformation file, Ecore files of source and target meta-models, source model file. Source and target models are expressed in XMI – XML format for UML.

5. Conclusions

We presented an approach for integrating two business process modeling notations – RAD and Hybrid IDEF using model mappings. A prototype system based on Eclipse platform was described. As future work we plan to complete the implementation and describe the mappings in a formal way.

References

[1] ECLIPSE Platform. `http://www.eclipse.org`.

[2] RADRunner. `http://www.rolemodellers.com`.

[3] AI0 WIN and ProSim tools. `http://www.kbsi.com`.

[4] Model-Driven Architecture. `http://www.omg.org/mda`.

[5] Idef0: Draft federal information processing standards publication 183. integration definition for function modelling. 1993.

[6] J. Bézivin. An introduction to the atlas model management architecture. Research Report LINA 05-01, 2005.

[7] C. Bădică et al. A new formal idef-based modelling of business processes. In Y. Manolopoulos and P. Spirakis, editors, *Proc. 1st Balkan Conference in Informatics BCI'03*, pages 535–549, Thessaloniki, Greece, 2003.

[8] C. Bădică et al. Role activity diagrams as finite state processes. In M. Paprzycki, editor, *Proc. 2nd International Symposium on Parallel Distributed Computing ISPDC'03*, pages 15–22, Ljubljana, Slovenia, 2003. IEEE Computer Society Press.

[9] K. Harrison-Broninski. *Human Interactions: The Heart and Soul of Business Process Management*. Meghan-Kiffer Press, 2005.

[10] R. J. Mayer et al. Information integration for concurrent engineering (iice): Idef3 process description capture method report. 1993.

[11] M. Ould. *Business Processes: Modeling and Analysis for Reengineering and Improvement*. John Wiley & Sons, 1995.