AgilPro –
Agile Processes in the context of ERP

Title of the document

AgilPro Metamodel description

Document information

last changes	version
14.02.2007	1.5

Document created by

Bernhard Bauer
Florian Lautenbacher
Stephan Roser

Address

Programming Distributed Systems Lab
Prof. Dr. Bernhard Bauer
Institute of Computer Science
University of Augsburg
Universitaetsstrasse 14
D-86135 Augsburg
Germany
phone: +49 821 598 2174
mail: bauer@informatik.uni-augsburg.de
www: www.ds-lab.org

Content

Images

1 Summary

This document describes the meta-model of the project AgilPro for agile processes in the context of ERP. This project is funded by the high-tech future offensive of Bavaria, Germany and fosters business process modeling in small and medium-sized enterprises (SMEs). Currently most companies only have proprietary software like DATEV, Sage, MS Word or Excel which can not be integrated easily. A process integration is not supported, only an integration of data is done using import and export functionality. This is the point where AgilPro fits in.

AgilPro is developed as a joint project between the University of Augsburg, Germany and eMundo GmbH, Unterhaching. The AgilPro tool suite consists of several applications: the AgilPro LiMo, the AgilPro Desktop applications, the Adapter and Integration framework and the AgilPro Enterprise Services.

The AgilPro LiMo is a tool for modeling business processes based on the Eclipse platform. It is an Eclipse RCP application where the model is based on a well-defined meta-model using the Eclipse Modeling Framework EMF. This meta-model supports the code generation and offers several views for different purposes: a business view for the manager, a technical view for the IT expert, an ISO 9000-view, an ITIL-view, etc. The model itself is drawn using the Eclipse Graphical Editing Framework (GEF). This document describes the underlying meta-model of AgilPro.

AgilPro will also be one of the cornerstones for the Eclipse Technology project Java Workflow Tooling (JWT). JWT focuses the usage of a workflow modeling tool which has a clearly defined API and can be rendered using several views. This is the Workflow Editing (WE) part of JWT. The WAM-part (Workflow Administration and Monitoring tools) focuses on the connection with a process engine, the deployment of the process, user administration, etc. For more information please refer to the website http://www.eclipse.org/jwt.

2 Meta-model of AgilPro: Abstract syntax & semantic

This section describes the meta-model of Eclipse Java Workflow Tooling (JWT) and the underlying A-gilPro contribution. The meta-model consists of several packages which are based on each other. The first one describes the graphical constraints whereas the latter ones are for the “real” meta-model concepts.

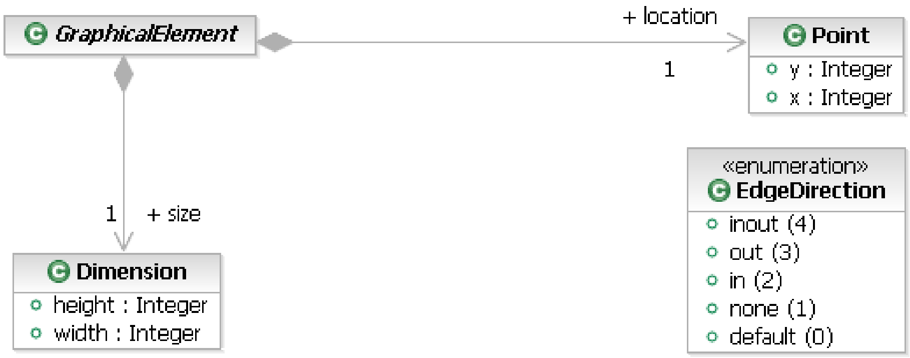


Figure 1: AgilPro – View

Each element which is visible in the graphical pane is a *GraphicalElement*. This has a location called *Point* with x and y value as well as a size (*Dimension*) specifying the width and height of the element. Additional there is the concept of an *EdgeDirection* which specifies whether an edge has arrows on one, both or none ends.

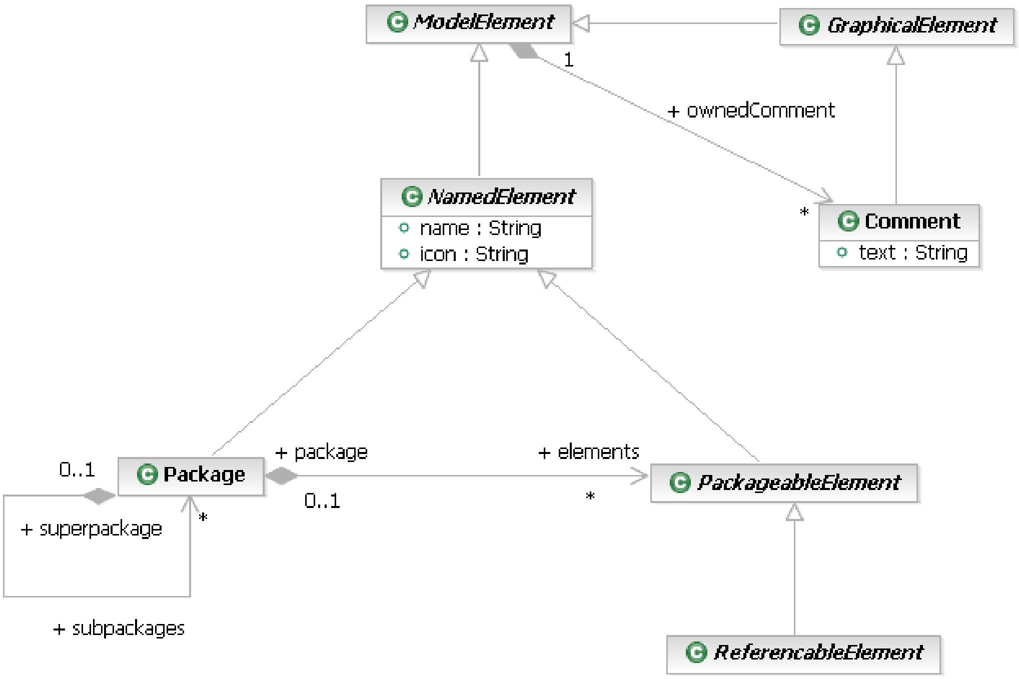


Figure 2: AgilPro – Core

Every element in JWT is a *ModelElement*. A *ModelElement* is the basic unit and the most abstract element of our meta-model. Every model element can have a textual *Comment*. A special kind of a model element is a *NamedElement*. All elements that have a name and optional an icon are at least *NamedElements*. A Package is a *NamedElement* and can have subpackages or other *PackageableElements*. This enables the user to structure his/her processes that belong to a specific area or to structure other elements that belong somehow together. A *ReferencableElement* is an element that can be packaged and referenced by other other elements (so called *References* introduced later).

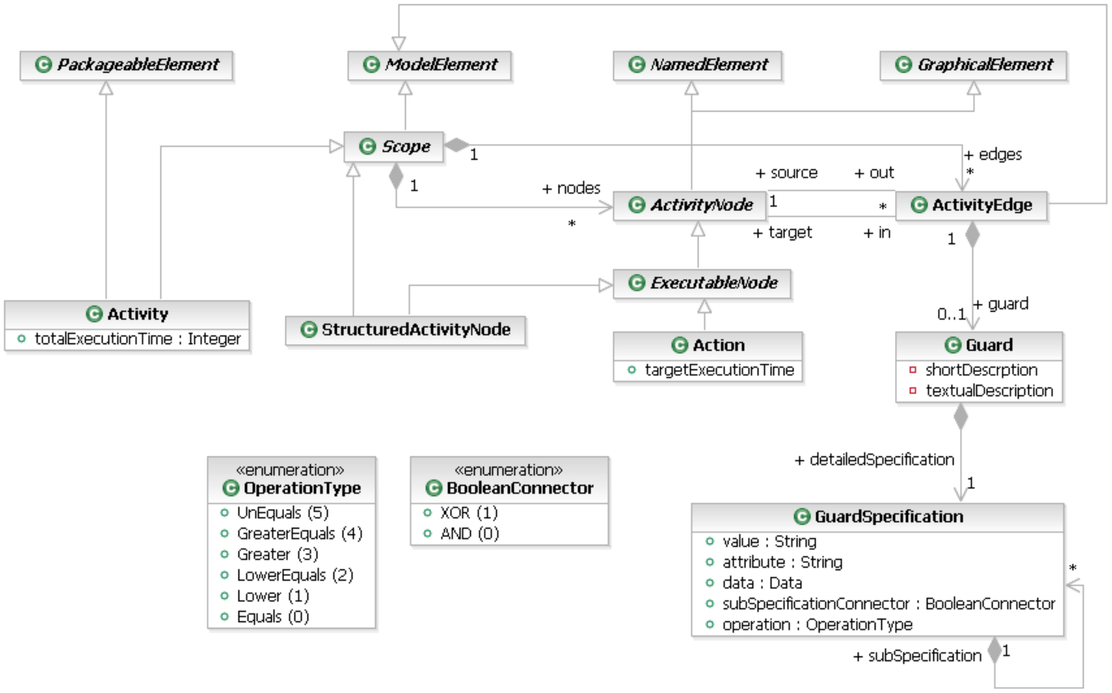


Figure 3: AgilPro – processes

All processes modelled with Eclipse JWT are *Activities*. An activity is a *PackageableElement* and can therefore be structured in packages. An *Activity* is a subclass of *Scope* which includes all elements in a graphical model. Examples for those elements are *ActivityNodes* and *ActivityEdges*. One example for an *ActivityNode* is an *Action* which is executable (subclass of *ExecutableNode*) and has a name and optional an icon (subclass of *NamedElement*). A *StructuredActivityNode* contains as an own scope itself *ActivityNodes* and *ActivityEdges*, but is itself executable from other nodes, too. Each *ActivityEdge* connects two *ActivityNodes* and might be constrained with a *Guard* which has a textualDescription and a more detailedDescription which can be simple Boolean terms (using the *OperationType*) or more complex terms connected through *BooleanConnectors*. Using the parameters of *Activity* (totalExecutionTime) and of all *Action*s (targetExecutionTime) one can simulate the duration of the process and compare it with the predefined value.

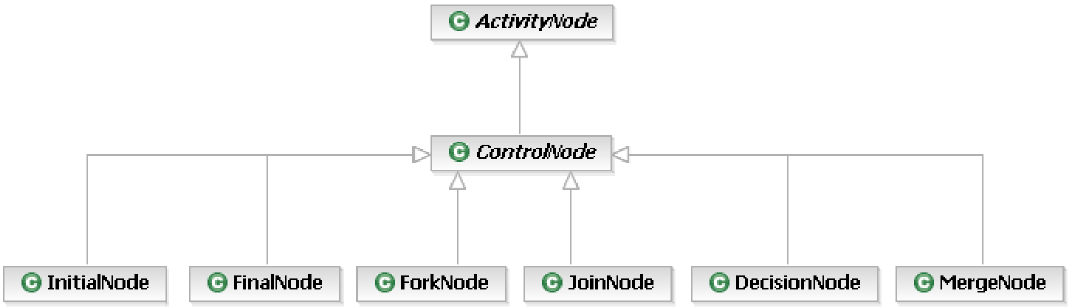


Figure 4: AgilPro – control nodes

To model the flow of several *ExecutableNode*s one can use *ControlNode*s. To model the start or finish of a process the *InitialNode* and *FinalNode* can be applied. To model parallel process flows and the synchronization afterwards one can use the *ForkNode* or the *JoinNode* respectively. For exclusive choices and merges afterwards the *DecisionNode* and *MergeNode* are available to the modeller.

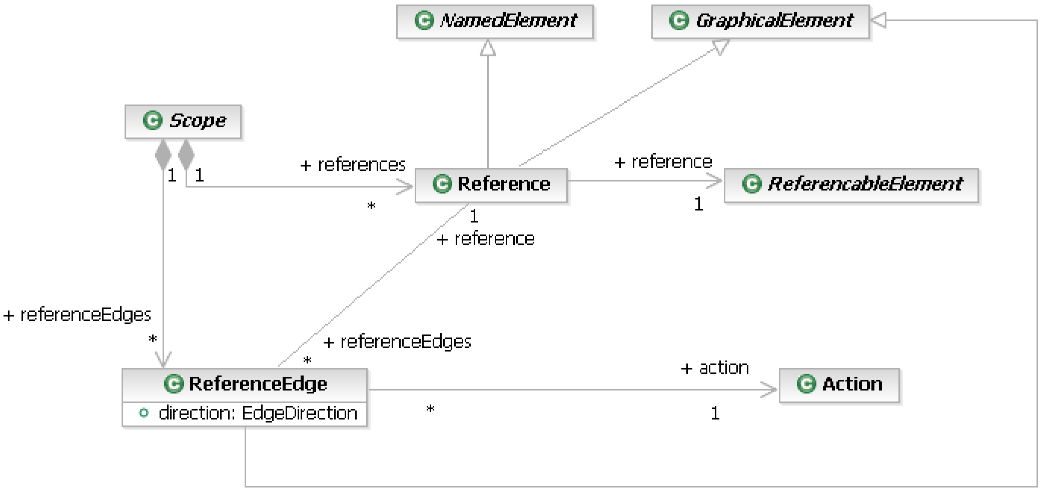


Figure 5: AgilPro – References

To include elements into the current activity that are normally outside the scope and defined for more than one process model, one can use the *Reference* to point to an existing *ReferenceableElement*. These References can be connected through *ReferenceEdges* with Actions. Example for a *ReferenceableElement* would be a *Role*, an *Application*, *Data*, etc. as shown later.

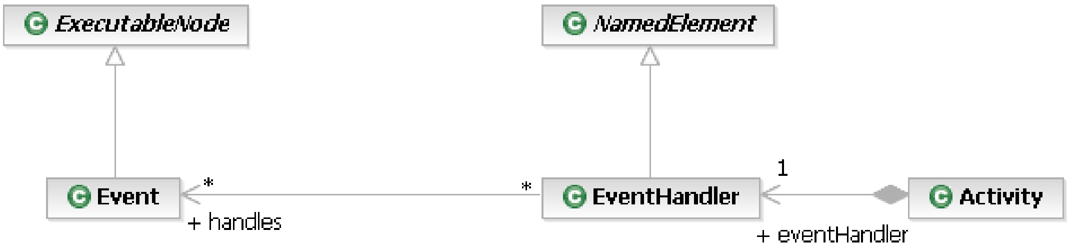


Figure 6: AgilPro – Events

To have the possibility to react to events from outside, one can include an *Event* into the process model. An *Event* is an *ExecutableNode* (similar to an *Action*). Each *Activity* includes an *EventHandler* who is responsible for the handling of an occurred *Event*. Such an event could be the arrival of a message, a time-out, etc.

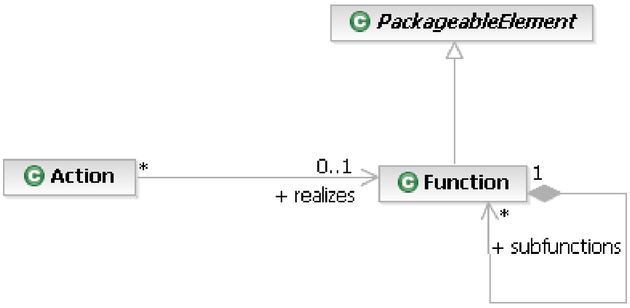


Figure 7: AgilPro – Functions

Each *Action* can be clustered into specific *Function*s. A function describes the kind of an action (e.g. Accounting). Each *Function* can be include in packages and might have several sub-functions belonging to itself.

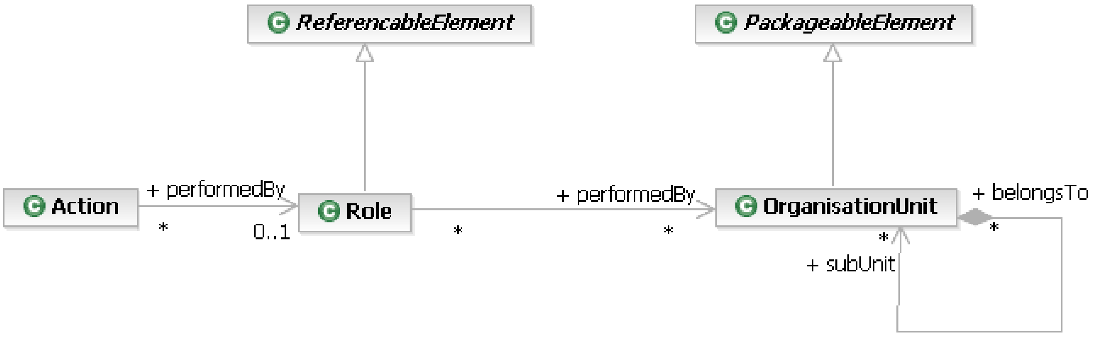


Figure 8: AgilPro – Organisations

Each *Action* can be performed either automatically or by a specific *Role* of an Organization. *Roles* are defined not only for one process model, but for all processes and are therefore *ReferencableElement*s. *Roles* can be grouped in *OrganisationUnits* which themselves can have sub units, too.

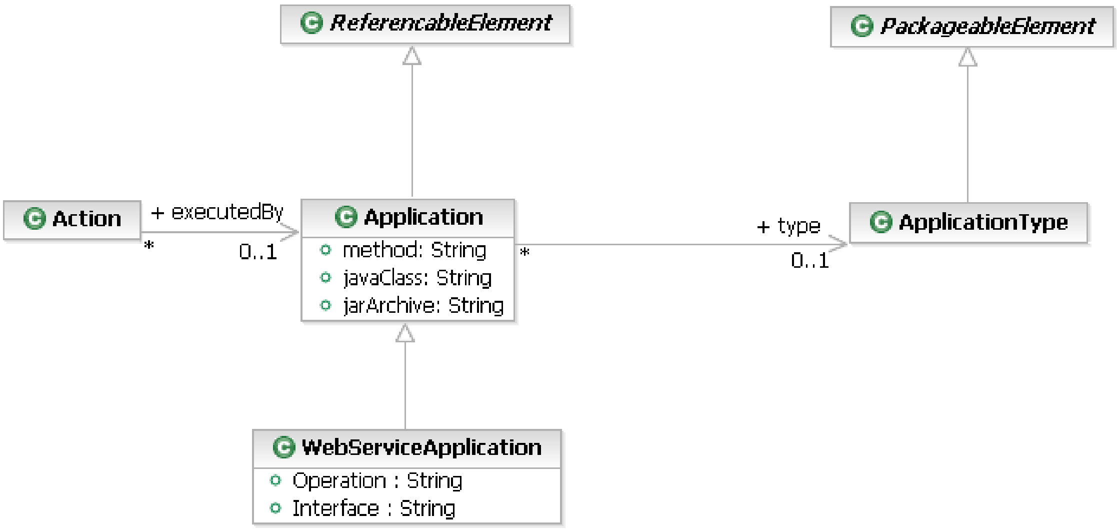


Figure 9: AgilPro – Applications

Each *Action* can be executed manually or alternatively by specific applications of the IT system. Again, *Application*s are defined for all kind of models and are therefore *ReferencableElement*s. Each *Application* can have an *ApplicationType* which clusters the applications. An application can be specified describing the javaClass and method which should be invocated and in which jarArchive this class is.

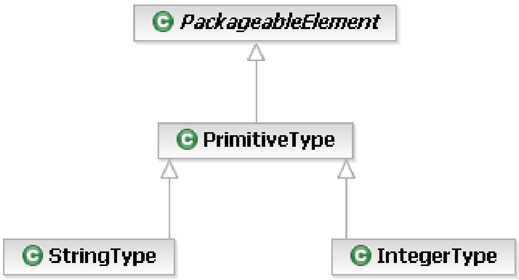


Figure 10: AgilPro – Primitive Types

An *Application* needs input and output data for its execution. These could either be *PrimitiveTypes* like textual *StringTypes* or numerical *IntegerTypes* or more complex types.

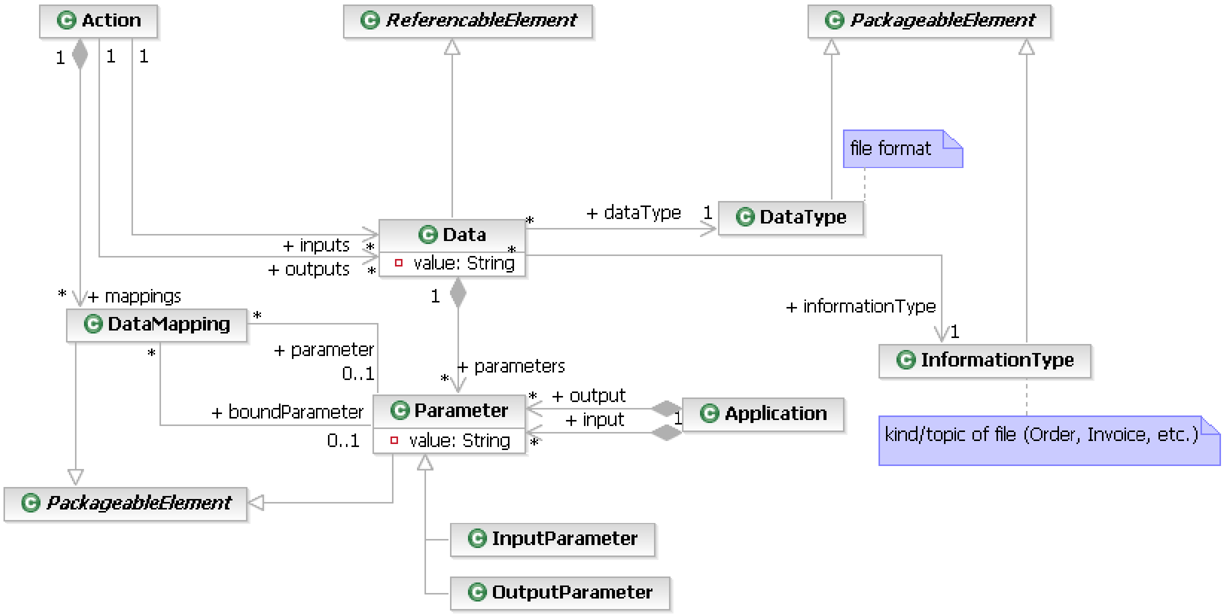


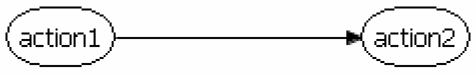
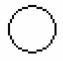
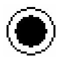
Figure 11: AgilPro – Data

Complex *Data* types can be described using their *DataType* which says something about the file format: is it a simple text file, an XML-file, an Excel sheet, a Word document, etc. On the other side it is possible to describe the *InformationType*, e.g. whether this is an order, an invoice, and so on. *Actions* either need these *Data* for their execution (inputs) or produce them after execution (outputs). Each action can consist of several parts, called parameters. Similar, applications can have parameters for their execution. To bind these parameters together the *DataMapping* exists which belongs to an *Action*.

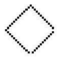
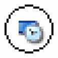
3 Meta-model of AgilPro: Concrete syntax

This section describes the concrete syntax of the AgilPro meta-model as it is displayed in the AgilPro LiMo (for Light Modeller).

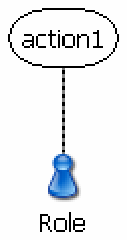
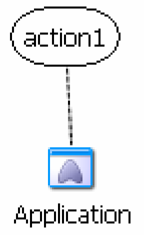
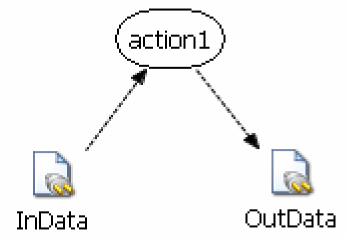
Actions are rounded rectangles which are connected through *ActivityEdges* with a filled arrow and a continuous line. *InitialNodes* are normal circles, *FinalNodes* have a filled circle in the outer circle.

		
Actions and ActivityEdges	InitialNode	FinalNode

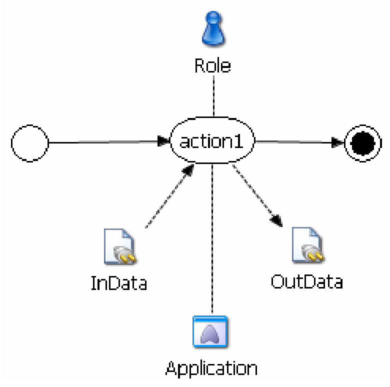
DecisionNodes and *MergeNodes* are diamonds, parallel flows and synchronization are modelled with *SplitNodes* and *JoinNodes* using a filled bar and Events are circles with a symbol inside that specifies the kind of the event.

		
DecisionNode / MergeNode	SplitNode / JoinNode	Event

The *Reference* to *Roles*, *Applications* and *Data* are symbolized with small icons and the name underneath. They are connected using *ReferenceEdges* which are symbolized with a dashed line. This line can have arrows for data elements specifying whether these are input or output data (or both, then the arrow is on both line ends).

		
Role	Application	Data

The following shows an *Activity* with one Action between an InitialNode and a FinalNode which is performed by the role Role, executed by one application and needs one input data and produces one output data.



Role
action1
InData
OutData
Application