ITEA2 – Project #11011

Multi-Concerns Interactions System Engineering

01.12.2012 to 31.03.2016

Traced process enactment prototype version 2

Project Deliverable D3.3.3

Task 3.3 –Advanced concepts in engineering process modelling and enactment: (lead UPMC)
WP3 – Advanced multi-concern engineering concepts (lead KU Leuven)

Status	[ ] Draft	**Document created**	: 21.03.2016
	[ ] To be reviewed	**Last edited**	:16.05.2016
	[X] Final	**Due date**	:29.02.2016
Confidentiality	[X] Public (for public distribution)	**Ready for review**	:18.03.2016
	[ ] Restricted (only MERgE internal use)	**Document version**	:0.1
	[ ] Confidential (only for individual partner(s))	**Pages**	:30

Contributors: Hakan Metin (UPMC), Reda Bendraou (UPMC) and Jacques Robin (UPMC)

Executive summary. This document provide a user guide for deliverable D3.3.3 which consist of two software delivered as plug-ins to the MERgE platform: (1) ProDAn 2.0 and (2) ProVer 1.0.

ProDAn 2.0 is the new, improved version of ProDAn 1.0 delivered as deliverable D3.2.1. It is an integrated plug-in to (a) model business processes using the standard fUML (Foundational subset for executable UML models), (b) guide their enactment but suggesting possible next actions by following the process model, (c) automatically detecting actions that deviate from this model, allowing such deviation to occur for a while and (d) automatically builds corrective plans to conclude enactment while minimizing the final deviation.

ProDAn 2.0 is based on the advanced concepts and experiments described in deliverable D3.3.2. It replaces the SAT solver technology underlying ProDAn 1.0 by the more scalable and expressive SMT solver technology.

The same solver technology and advanced concepts are also used in ProVer 1.0 which complements the business process engineering services integrated in ProDAn. ProVer allows a user to specify desirable properties that a process model must possess and automatically verifies them prior to its enactment. It thus supports prevention of many enactment deviations due to design-time problems. It allows a user to specify properties of various classes: soundness, resource constraints and business rules. Soundness properties include termination along all paths, termination along at least one path, termination along a given path and reachability the final process node. The resource constraint properties include constraints on execution time, manpower for different roles required for different activity classes. The business rule constraints include arbitrary constraints specific to the process model domain such as a limit on the number of time an activity of a given class must be executed.

If overdue, provide reason here: the software code was available on time. We had slightly underestimated the time that it will take us to write the documentation.

Version.	Content	Resp. Partner	Date
1.0	User guide and software	UPMC	17/03/2016

Reviewed & Accepted	Name	Partner
Independent Reviewer (outside WP)	Henri Lindberg	nSense
Validation from Management Board	C. Robinson	THALES R&T

Contents

List of Figures

1 User Guide ProDan

1.1 Overview

PRODAN is a software process enactment tool that allows enacting the process models in an environment where the modeller is guided throughout the execution to allow temporary process deviation while minimizing final deviation at the end of the enactment. It automatically detects deviations, warns the user of its occurrence and then suggests actions to recover from it. Different approaches follow different techniques for handling process deviations. Some tools restrict users (e.g. project managers) to execute the activities only in the order that they appear in the original process model, while others ignore process deviations completely.

PRODAN analyses the user's decision for activity execution and detects if it would result in a process deviation. If this is the case, PRODAN notifies the user showing the consequences in terms of missing and subsequently available activities. At any point after selecting a deviating path, the user can ask PRODAN for a recovery path, i.e., a list of missing action towards concluding the enactment in best possible conformance to the process model. In all cases the tool helps the user make informed decisions.

PRODAN, as a process enactment tool can be used by project managers to enact process models. This enactment of process models simulates the control flow and data flow perspective of the real life processes, carried out in the enterprises. PRODAN helps in developing the understanding of the flows of a process, before its actual implementation. PRODAN can also be used by the process modellers to analyze the effect of different deviations that can occur during the enactment of the process. The control flow behaviour of a process deviation can be analyzed by the suggested continuation options offered by tool, after the deviation. This helps to guides project managers to make informed decisions for the enactment of processes.

PRODAN is a plug-in to the MERgE multi-concern system engineering platform It is open source and is distributed under Eclipse Public Licence EPL.

1.2 Plug-in Installation

A prerequisite for installing PRODAN V2.0 is the installation of MERgE platform version 2.x onwards. Two mechanisms are offered for installing the plug-in into the MERgE platform. It can be installed via an update site or by using the plug-in archive provided alongside this user guide.

Though the archive content can be copied manually into the MERgE platform plugins folder, we strongly recommend to use one of the two suggested approaches for the installation of PRODAN plug-in.

1.2.1 Installation - update site

1. Open MERgE platform (Eclipse), and go to `Help` → `Install New Software`.
2. This shall open an install window. Click on the `Add` button. This shall open an Add Repository dialog box. Fill it with the following information and click on `OK`:

Name : *PRODAN*

Location : *http://merge.lip6.fr/MERGE_PRODAN*

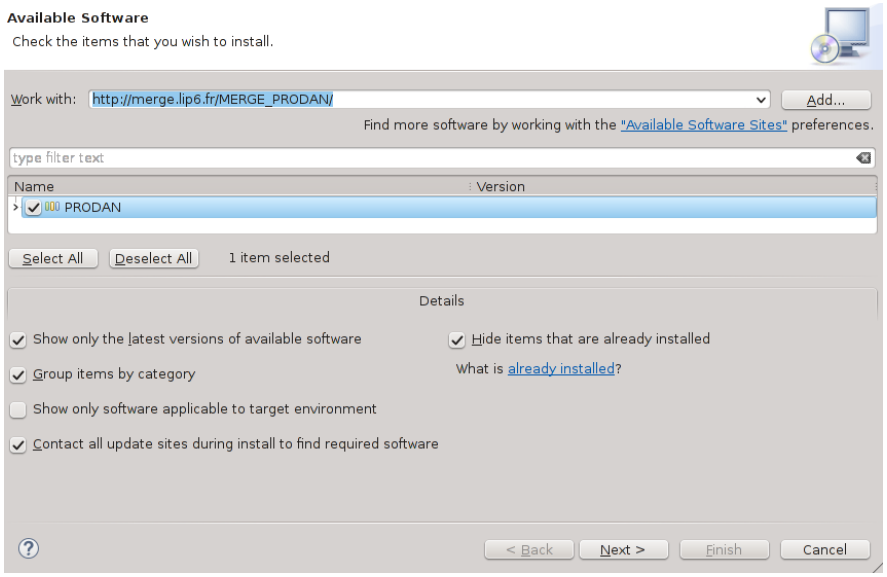

Figure 1 ProDan update site installation

3. Select the plug-in `PRODAN` and click on `Next` (Figure 1 ProDan update site installation). Click on `Next` once again for the installation details.
4. Accept the license agreement and click `Finish`.

1.2.2 Installation - Local archive

1. This user guide is distributed alongside the plug-in and an example process model. The plug-in is archived in a file named `PRODAN`.
2. Open the MERgE platform, and go to `Help` → `Install New Software`.

3. This shall open up a window for plug-in installation. Click the `Add` button. This shall open an Add Repository dialog box. Add the name of the plug-in as PRODAN.

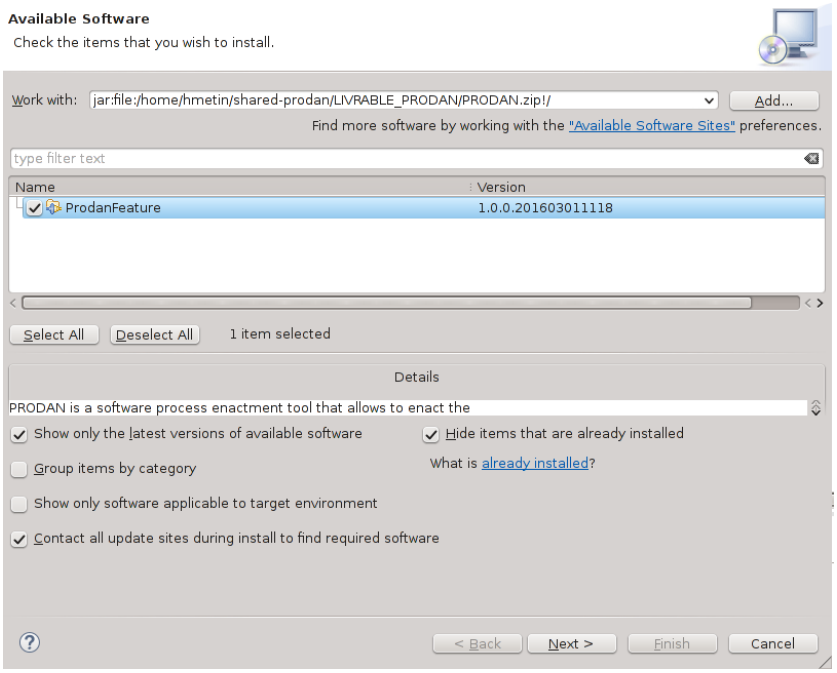

Figure 2 ProDan archive installation

4. Click `Archive` and browse to the zip file.
5. Click `OK` and uncheck the option `Group items by category`.
6. Select the plug-in `ProdanFeature` and click on `Next`. Click on `Next` once again for the installation details.(Figure 2 ProDan archive installation)
7. Accept the license agreement and click `Finish`.

1.3 Tool Layout

PRODAN is deployed as an eclipse plugin, it is composed of different views. Each of these views is discussed in detail in turn in what follows.

1.3.1 Process Editor

The process editor allows the development of process models using UML 2.4 activity diagrams. ProDAn relies on the control flow of a process model to detect process deviations. The editor allows adding activities, actions, initial node, final node, and other control flow nodes like decision, merge, fork and join nodes. Apart from these nodes, the `activity tools` toolbox at the right side also allows adding control flows between these nodes. Figure 3: Process editor view shows a testing sub-process that is a part of a complete software development process. It contains four activities: *Prepare release candidate*, *Regression test*, *Ad hoc testing* and *Release for user acceptance test*. This process occurs in the final stage of testing after integration and system testing and before user acceptance testing.

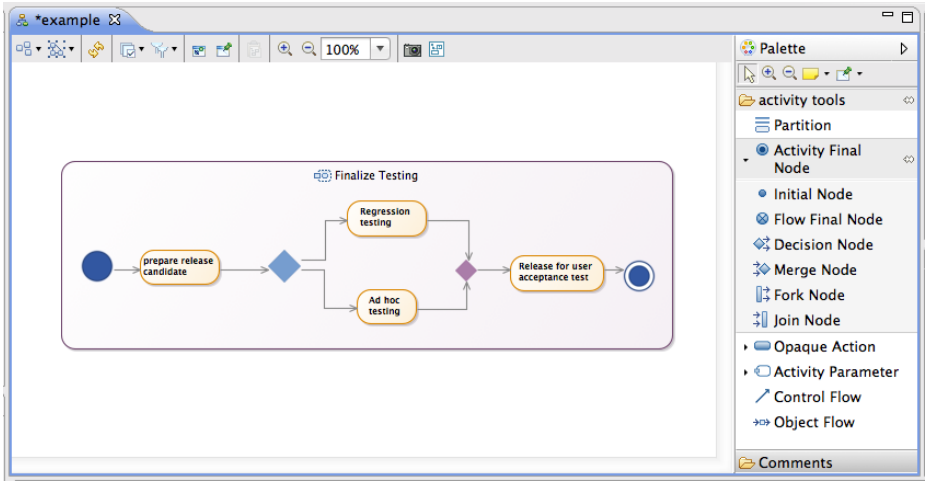

Figure 3: Process editor view

ProDAn uses UML Opaque Actions to model atomic activities of a process model that cannot be decomposed into sub activities. The description of an opaque action describes its inner implementation. It can be added to the action using its "body" property under the semantics tab of properties view. Different roles associated with the process model can be modelled using partitions in activity diagram, which is a similar concept as swim lanes in BPMN.

1.3.2 Process Enactment view

ProDAn's enactment view allows enacting a modelled process. It provides controls to start and stop the enactment. Once the execution of the process model is started, this view lists all the activities that can be executed in the process model.

This view also shows whether an activity is currently be executed. Figure 4 shows the enactment view for the example process of Figure 4 Process enactment view after the execution of the *Prepare*

release candidate activity has already completed its execution. The view suggests the user to execute any of the two activities: *Regression testing* or *Ad hoc testing*.

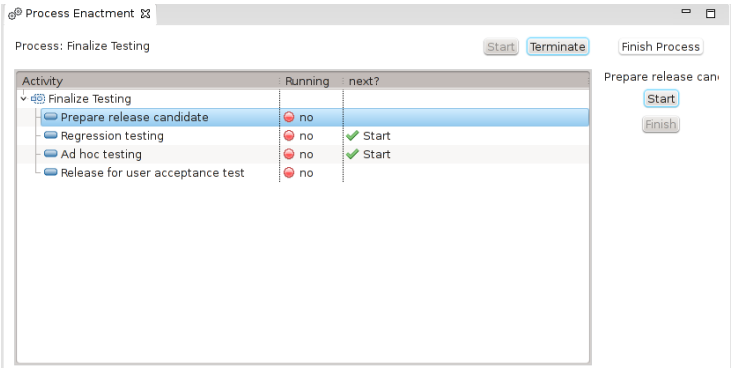

Figure 4 Process enactment view

Apart from presenting the list of executable activities and their current state, this view also shows to the user the set of possible activities to execute next. This set is computed by the underlying SMT constraint solver based on the trace of the activities enacted so far. It can thus include activities, which were skipped due to past deviations from the process model in this enactment trace. A deviation is defined as a constraint violation. When the user presses the Finish Process button, the enactment view provides a recovery plan if the enactment trace contains a deviation from the process model. This plan is a list of activities that must be executed to complete the enactment in conformance to the process model. Note that in some cases, the plan does not lead to full process model conformance because the deviation strayed too much from it.

1.3.3 Execution Trace

The process execution trace view shows a trace of the process model enactment, listing all the executed activities in order. Figure 5: Execution trace view shows the trace view of our process example, where *Regression testing* activity was executed after the *Prepare release candidate* activity.

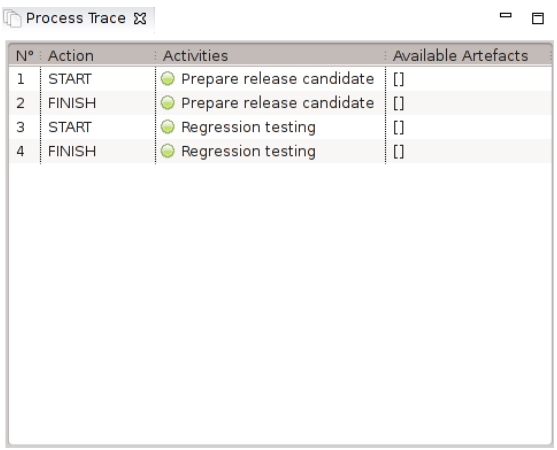

Figure 5: Execution trace view

1.4 Getting started by example

Let us try to execute a simple example process shipped with this tool to help understand the process enactment, deviation and recovery aspects of the tool.

1.4.1 Setup the environment and example

The following steps can be used to setup the environment for process enactment.

1. Open the MERgE platform (or Eclipse) and select `File` → `Import`. Then select `General` → `Existing Projects into Workspace` and click `Next`. (see Figure 6 Import UML Project)
2. Select `archive file` option and browse to the example archive file, *Finalize Testing Example.zip*, provided alongside this plugin. Click `Finish`.

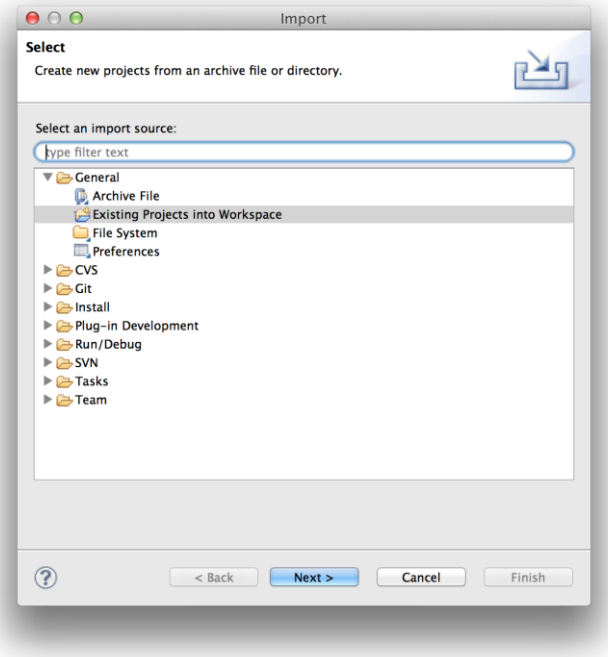

Figure 6 Import UML Project

3. This will import a UML project in your current workspace. Open it in the explorer view. Double click the *example* diagram to open it in the editor. (see Figure 7 Expand Model explorer)

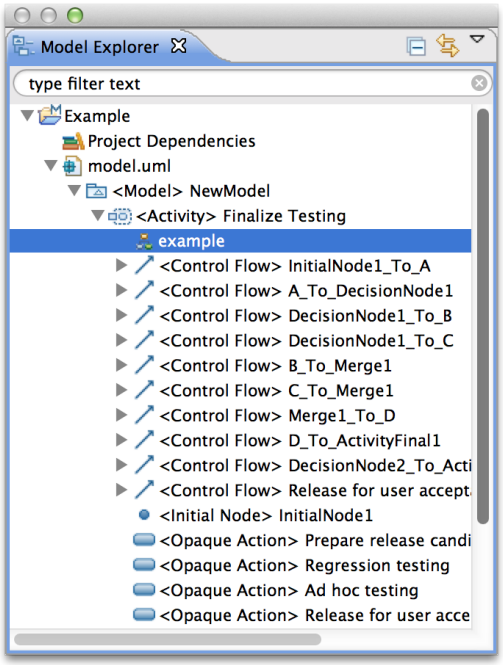

Figure 7 Expand Model explorer

4. This shall open up theUML activity diagram in the editor, as shown in Figure 8 Example Process model in editor

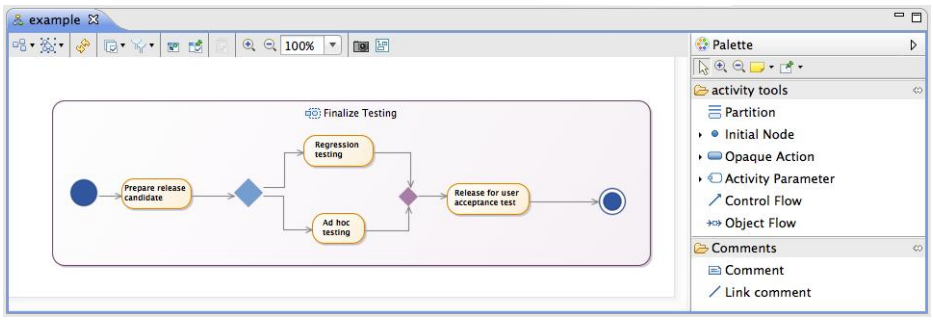

Figure 8 Example Process model in editor

5. Select `Window` → `Show view` → `Process Guidance` to open up Process Enactment, Trace, Guidance.

1.4.2 Execute example process

You can follow the following steps to execute the *Finalize Testing* example:

1. Select the *Finalize Testing* process in the editor and click `Start` to initiate the execution of the process. It takes a little time to load the process model and when it is ready, you then see a list of all the process model activities in the process enactment view, and can notice that none is running at the moment. Even though the process had been initiated at this stage, no particular activity started its execution yet. *Prepare release candidate* is the first activity proposed by ProDan.
2. Click on *Prepare release candidate* either from the list in the enactment view or on the node for this activity in the diagram displayed by the model editor and click `start`. This executes the activity and changes its status changes in the enactment view.
3. Once an activity has been started by clicking on the Start button, you can finish it by clicking on the Finish button. If you do so for the Prepare release candidate, ProDAn proposes to continue enactment with either the Regression testing activity or the Ad hoc testing activity, as shown in Figure 9 Enactment view - Termination of *Prepare release candidate*.

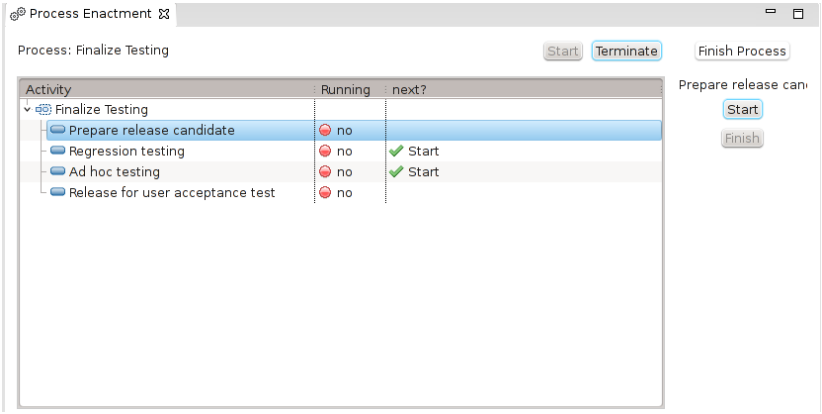

Figure 9 Enactment view - Termination of *Prepare release candidate*

4. Start and finish *Regression testing*. Then start *Ad hoc testing*. When you try to execute *Ad hoc testing*, a message box pops up that shows a deviation occured if you continue with this execution, as shown in Figure 10 Warning message for Deviation Continue the execution anyway.

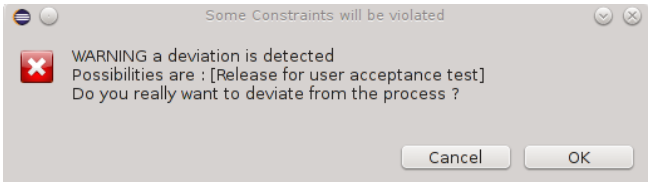

Figure 10 Warning message for Deviation

5. The `Trace view` can be consulted to see which activities have been executed so far and in what order. Resulting from our execution scenario, we have *Prepare release candidate*, then *Regression testing* followed by *Ad hoc testing* activities respectively, as shown in Figure 11 Trace After Deviation. Then *prepare release candidate* was executed. You can notice red and green dots . In this view, a red dot to the left of an activity name indicates that the execution of this instance of the activity introduced a deviation from the process model. In contrast, a green dot indicates that this execution conformed to the process model.

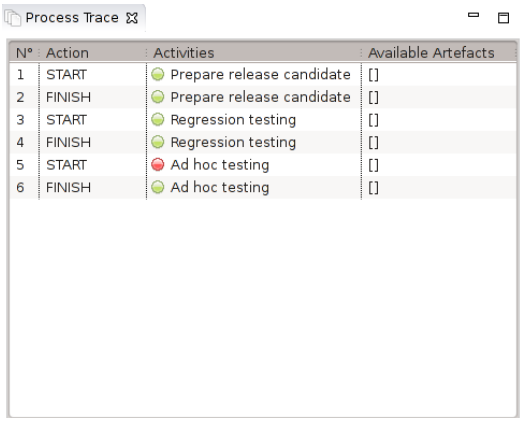

Figure 11 Trace After Deviation

6. As this point clicking `Finish Process` button, opens a message box pop up that proposes a recovery plan to conclude the enactment in a state in conformance with the process model. This is shown in Figure 12 Recovery plan.

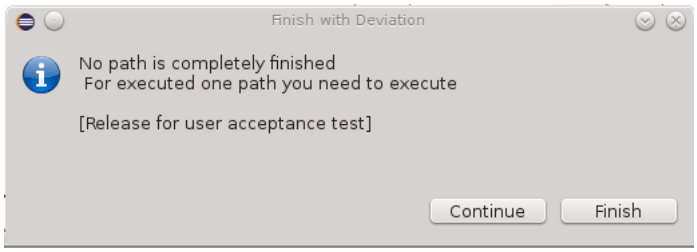

Figure 12 Recovery plan

7. At this point you can either finish enactment in conformance with the process model by executing the proposed recovery plan, or in deviation from the process model by ignoring the recovery plan.

1.5 Process modelling

PRODAN uses Obeo UML Designer to edit process models prior to their enactment. Application of specific profiles developed for the PRODAN process models allows adding specific attributes to them.

1.5.1 Creating a new process model

1. In the MERgE platform, go to `File` → `New` → `UML Project`. Then give this project a name e.g "Sales Example". This will create a new project in your workspace. Click `Finish`.
2. Use `Window` → `Show view` → `Process Guidance` to open the process enactment and trace views. This will automatically create a UML project called model.uml.
3. In the `Model explorer view`, open the project hierarchy for this project, and then right click `NewModel` → `New Representation` → `Activity Diagram`. Give it a name like "Sales Activity Diagram". (see Figure 13 Creating an activity diagram)

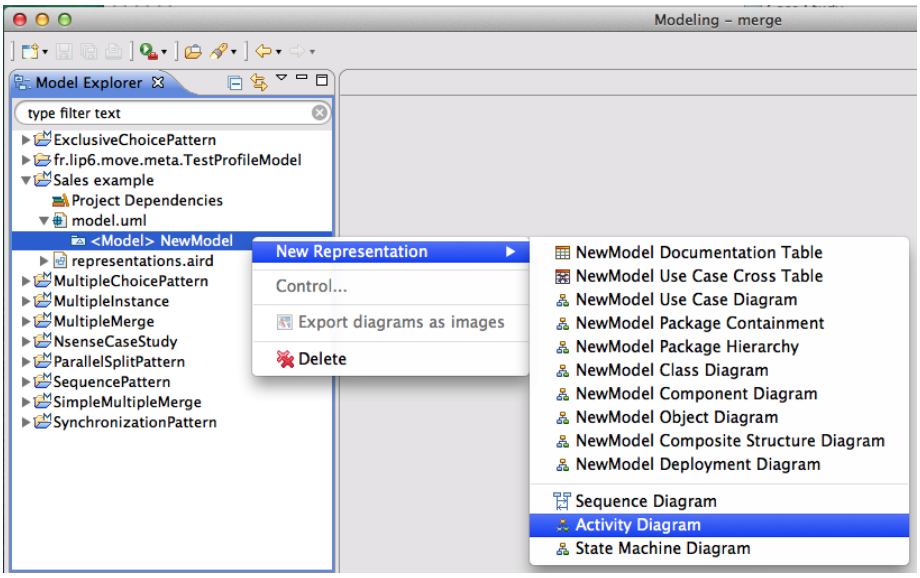

Figure 13 Creating an activity diagram

4. Select the NewModel in the process editor and enlarge it so that multiple activities can be placed inside it. Go to the `properties view` and change its name to *Sales Process*.
5. Choose `Partition` from the activity tools at the right side and create two partitions in this model. Using `Properties view`, change their names to *Sales department* and *Warehouse*. Place an Initial Node and an Activity Final Node outside the partitions, as shown in Figure 14 Creating partitions for different roles.

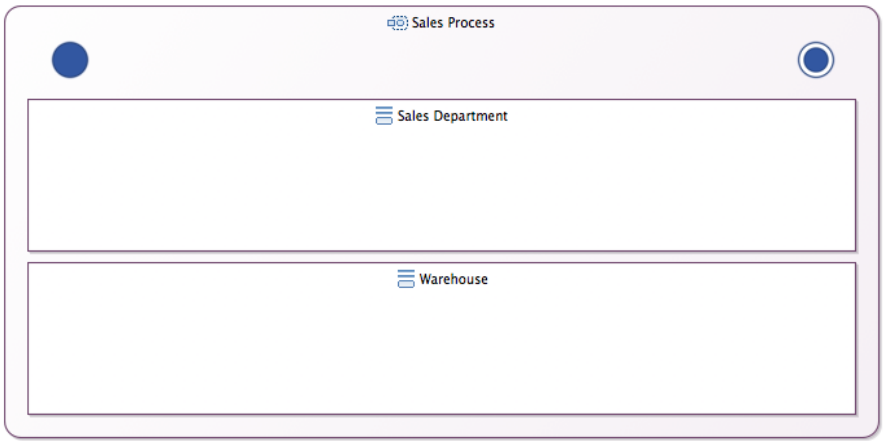

Figure 14 Creating partitions for different roles

6. Now create opaque actions inside the partitions. You can change now the name of the activities from `properties view` → `General` and remove the body text from `Properties` → `Semantic` → `Body`. You can also resize the activities
7. From the activity tools, select `decision node`, `merge node`, `fork node` and `join node` and place them in the process model as shown in Figure 15 Complete Sales Process. Using the `control flow` link the *initial node* to *Receive order* activity, the Receive order node to first decision node and so forth until linking the Ship order node to the Activity final node.

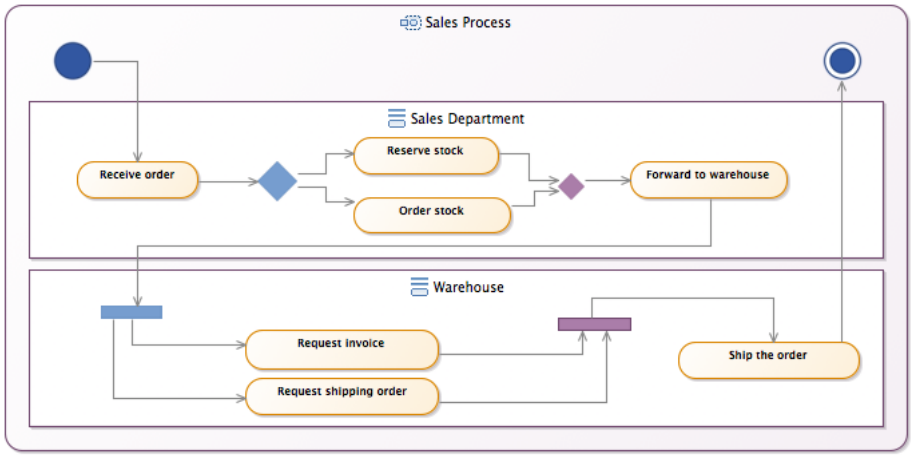

Figure 15 Complete Sales Process

1.6 Process Enactment

The execution of a process model is called its enactment, in the process domain. We shall execute the process model that we developed in the earlier section, the *Sales Example* process.

1.6.1 Starting and terminating a process

1. From the *Sales example* UML project, open the *Sales Activity Diagram* in the editor view.
2. Open the `process enactment view`. The top row of this view shall state that process is not selected. We select the *Sales Process* from the `editor view`, so that it can be loaded in this view.
3. Once it is loaded, we can see three buttons across the process name: `Start`, `Terminate` and `Finish Process`. `Start` button shall The Start and Terminate buttons respectively start and terminate

enactment, while the `Finish Process` requests Prodan to compute recovery plans following a deviation.

4. Once the process is executed, we can see its activities in the table shown in the `process enactment view`.
5. Once `process enactment view` is populated with activities , we can continue the execution of individual activities.

After the execution of all the activities, one should terminate the execution of the process.

1.6.2 Starting and completing an activity

1. Once the process is started, the user can execute its activities. In our example process model, because Receive Order the only activity that immediately follows the initial node, Prodan guides the user by indicating this activity as the only possibility to start enactment in conformance to the process model. Starting with any other activity would constitute a deviation(Figure 16 Enactment view - Start of Sales Process).

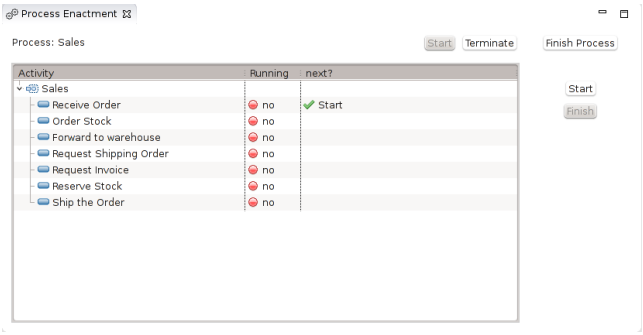

Figure 16 Enactment view - Start of Sales Process

2. We start with executing the first activity i.e. *Receive order*. This activity can either be selected from diagram or from the process enactment view. Once the activity is selected, the start button above the table on the right can be used to start its execution. During the execution of the activity, we can see that it is running from the `process enactment view`. The running state of the activity is enabled in the table. (see Figure 17 Enactment view - Running *Receive order* activity)

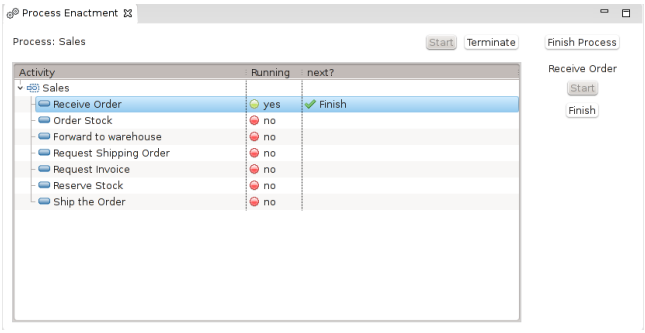

Figure 17 Enactment view - Running *Receive order* activity

3. Now the `Finish` button in the process enactment view can be used to stop the execution of this activity. After the completion of *Receive order* activity, we can see that no activity is running in the `process enactment view`. However there are two suggested activities to be executed next: *Reserve order* and *Order stock*. (see Figure 18 Enactment view - Two suggested activities)

Suggestion of multiple activities in process enactment view indicates mutually exclusive alternative options that can constitute a valid next step in conformance to the process model.

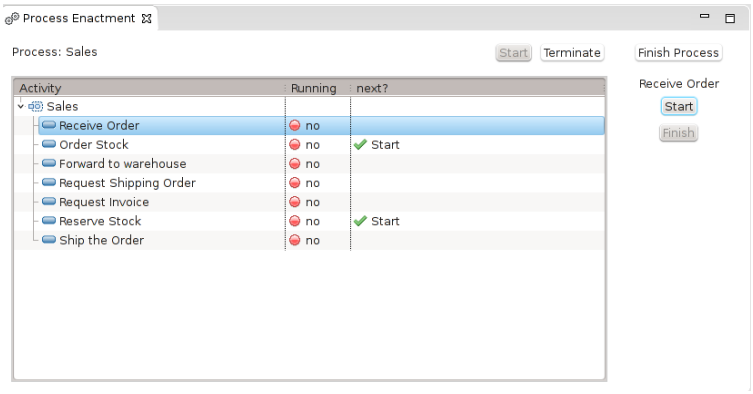

Figure 18 Enactment view - Two suggested activities

4. The user can continue the execution of activities by following the suggested activities by the tool. This would ensure that process does not deviate from the original planning. Once the final activity is finished, the process can be terminated.

1.7 Understanding process trace

Let us follow the steps 1 - 3 of section Starting and completing an activity, this gives us the state where the process is being enacted and only *Receive order* activity was started and finished by the user. After *Receive order* activity, we can choose any of the two activities *Reserve stock* and *Order stock*. Let us execute *Order stock* activity and then finish it. We can continue further to the execution of *Forward to warehouse* activity.

At this point if we open the `Trace view`, we can see the execution trace for the process enactment of *Sales Process*, as shown in. This trace gives us the order in which the activities were executed up to the current state. Thus we can see that the execution started from *Receive order* activity, followed by *Order stock* activity and finally *Forward to warehouse* activity was executed.

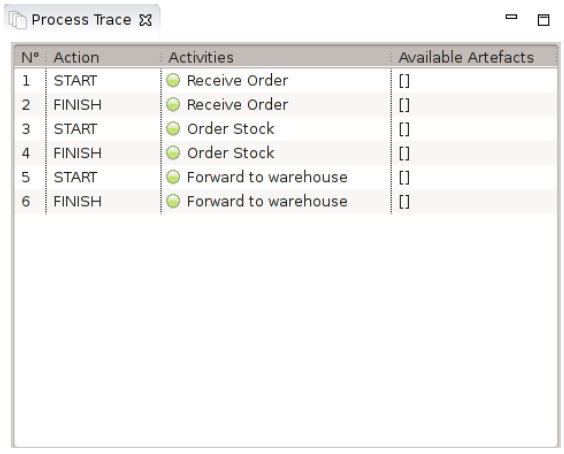

Figure 19 Process enactment trace

1.8 Process Deviations

1.8.1 Deviating from a process

1. Let us follow the steps 1 - 3 of section Starting and completing an activity, this gives us the state where the process being enacted and only *Receive order* activity was executed and finished by the user. After *Receive order* activity, we can choose only one of the two activities *Reserve stock* and *Order stock* to remain in conformance to the process model.
2. Execute *Order stock* activity and then finish it. At this point, the tool suggests us to execute *Forward to warehouse* activity. Executing Reserve stock now would constitute a deviation from the process model violating the mutual exclusion constraints of alternative activities following a given decision node.
3. Trying to execute the *Reserve stock* activity at this point triggers a warning message pop-up that this choice constitutes a deviation from the process model, as show in Figure 20 Warning message box for deviation. The user can choose to cancel the chosen execution by clicking on the cancel button. Or the user can go ahead with in spite of the warning to intentionally deviate from the process model by clicking on the OK button. The resulting deviated enactment trace view is shown in Figure 21 Trace view after deviation. The actions marked with a red dot in this view indicates the actions that constituted a deviation.

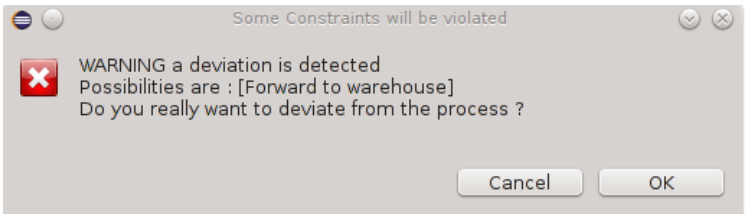

Figure 20 Warning message box for deviation

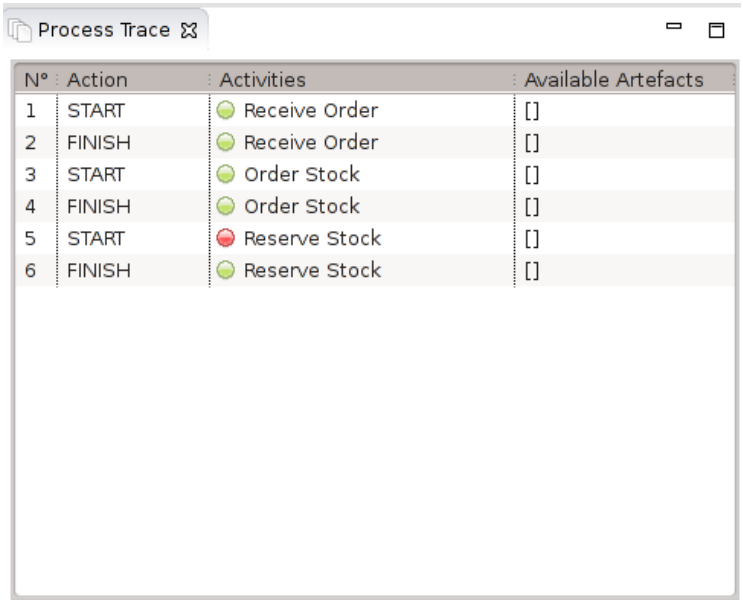

Figure 21 Trace view after deviation

1.9 Recovery Plan

In the deviated enactment state shown in Figure 21 Trace view after deviation, if we click on the Finish Process button, Prodan will pop the modal window shown in Figure 22 Recovery Plan. It gives two possible recovery plans, i.e., sequences of activities to execute in order, to finish enactment back to conformance with the process model. Clicking on the Finish button terminates enactment in the current deviated state. Clicking on the Continue button, allows resuming enactment and possibly execute one of the recovery plans computed by Prodan.

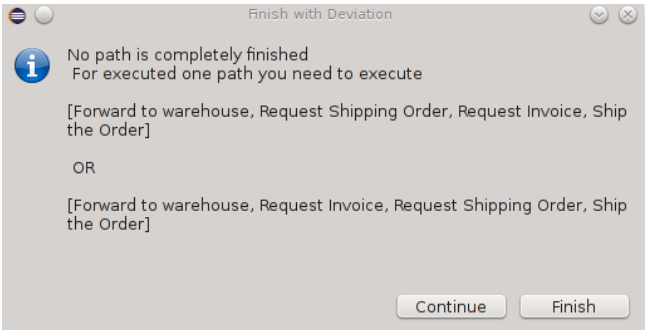

Figure 22 Recovery Plan

2 User Guide ProVer

2.1 Overview

Investigations on some process repositories demonstrated that between 10% [Mendling 09] to 50% [Vanhatalo 07] of process models are flawed and contain inconsistencies. In some critical businesses, having unsound process models could be very harmful for the quality of the delivered services and products. Inconsistencies can range from very basic syntactical errors, to more complex issues such as deadlocks, inconsistent allocation of resources and time duration on process's activities or any proprietary business constraints that might be violated by potential process executions (paths). Some process models, depending on the business domain (healthcare, military, etc.), can be very complex and may contain more than 250 activities with very sophisticated control and data flows, resource and time constraints [Christov 14] [Simidchieva 10]. Trying to analyze these processes and to verify them without the help of a tool would be unmanageable. In [Mendling 09], a study demonstrated that over 2000 inspected process models, 10% of these models where unsound. An equivalent study on SAP repository containing more than 600 complex process models demonstrated that more than 20% had flaws. [Mendling 06, 07]. Similarly, Gruhn et al. [Gruhn 07] collected 285 EPC (Even-Driven Process Chains) from different sources i.e., repositories, scientific papers, thesis, etc., and came to the conclusion that even though process models were syntactically correct, more than 38% of them were unsound. Finally, Vanhatalo et al. [Vanhatalo 07] analyzed more than 340 Business process models with a focus and the control flow aspect to realize that half of them were invalid i.e., contained deadlocks or unreachable activities. To avoid this kind of errors, formal verification on these models is needed. Properties implanted in ProVer allows anyone to verify his model very easily by simply choose properties on ProVer graphical mode.

ProVer can be used to verify three classes of properties: soundness properties, resource constraint properties and business rule properties. The soundness properties include termination along all paths, termination along at least one path, termination along a given path and reachability of a, ActivityFinal node. The resource constraint properties include constraints on execution time, manpower for different roles required for different activity classes. The business rule constraints include arbitrary constraints specific to the process model domain such as a limit on the number of time an activity of a given class must be executed.

ProVer is an eclipse plugin which use an SMT solver which is z3 (Microsoft Research MIT License) This tool uses UML Designer by OBEO to visualize process diagrams. User simply chooses the properties to be verified and launch the verification.

ProVer can be integrated to the MERgE multi-concern system engineering platform. It is open source and distributes under EPL (Eclipse Public Licence)

2.2 Properties implanted on ProVer

A categorization of the different properties that can be expressed on software process models is presented on the following table. It represents the outcome of a literature review in the business process domain and in software methods and practices.

Category	Definition
(1) Soundness	
OptionToComplete	A started process can always complete
ProperCompletion	No other activity should be running when the process terminates
NoDeadTransition	All the activities must be reachable
Soundness with data	
MissingData	The data are always present when they need to be accessed *(e.g. no data missing to start an activity)*
UselessData	The data created are always used *(e.g. no data created but never used before the process ends)*
InconsistentData	The data can never be in an inconsistent state *(e.g. no data modified by multiple activities in parallel)*
(2) Organizational	
InTime	There is enough time to perform the activities *(e.g. the process will terminate before X hours/days)*
MissingResource	No missing resource to start an activity *(e.g. there are enough agents to do the process)*
(3) Business	
ExistenceActivity	`A` is executed more/less/(between) X (and Y) times
ExistenceTimeActivity	`A` is executed before/after/(between) `X` (and `Y`) time unit
ExistenceTimeData	`ArtefactA` is available before/after/(between) X (and `Y`) time unit
ExistenceTimeResource	`ResourceA` is used before/after/(between) X (and `Y`) time unit
Relation	`A` is executed before/after/in-parallel/in-exclusion/(between) `B` (and `C`)
RelationData	`ArtefactA` is available before/after/in-exclusion of `ArtefactB`
RelationActivityData	`ArtefactA` is available before/after/in-parallel/in-exclusion/(between) the execution of `B` (and `C`)
...	...

Figure 23 Overview of the software properties

2.3 Plug-in Installation

A prerequisite for installing ProVer is the installation of MERgE platform 3.0.1. Two mechanisms are offered to install the plug-in on MERgE platform.

2.3.1 Installation - Update Site

1. Open MERgE platform and go to `Help` → `Install New Software...`
2. This shall open a new window as shown in figure . Click on the `Add...` Button. This shall open an Add Repository dialog box. Add the following informations and Click on `OK`.

Name : *ProVer*

Location : *http://merge.lip6.fr/MERGE_PROVER*

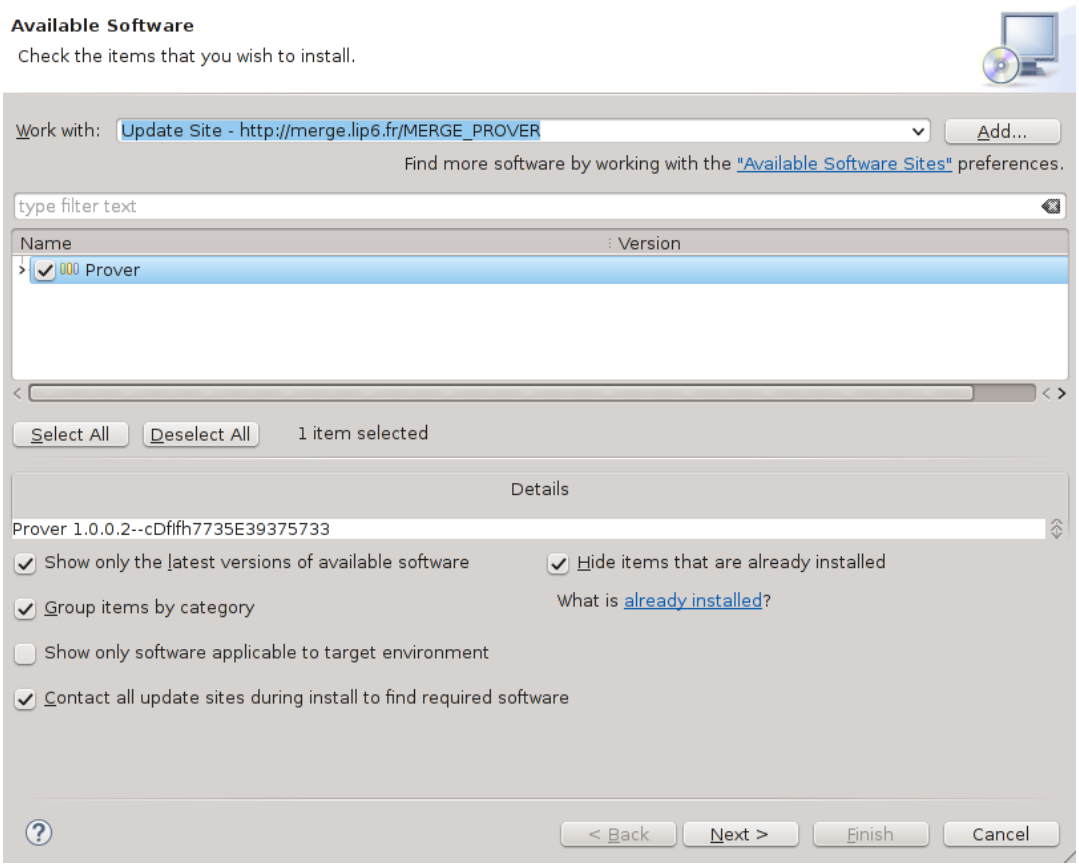

Figure 24 Prover update site installation

3. Select the plug-in `Prover` and Click on `Next`. Click on `Next` once again for the installation details. (Figure 24 Prover update site installation)
4. Accept the license agreement and Click on `Finish`.

2.3.2 Installation - Local Archive

1. The plug-in is archived in a file named ProVer.zip
2. Open MERgE platform, and go `Help` → `Install New Software`, this shall open up a window for plug-in installation

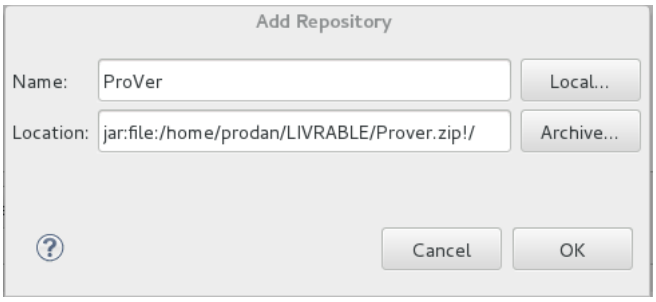

Figure 25: Installing the plug-in - archive file selection

3. Click on `Add`, this shall open an `Add Repository` dialog box.

4. Click on Archive and browse ProVer archive File (see Figure 25: Installing the plug-in - archive file selection) and click on `OK`
5. Uncheck the option `Group by category`
6. Select the plug-in and click on `Next`, click in `Next` once again for the installation details.
7. Accept the license agreement and Click on `Finish`.

2.4 Tool Layout

ProVer is deployed as an Eclipse plug-in, It is composed of different views. Each of these is detailed in the following:

2.4.1 Process Editor

The used process editor is `UML Designer` provide by Obeo. It allows the development of process models using UML 2.4 activity diagram. The editor allows adding activities, actions, initial node, final node, and other control flow nodes like decision, merge, fork and join nodes.

ProVer uses UML Opaque Actions to model atomic activities of a pro- cess model that can not be decomposed into sub activities. The description of an opaque action describes its inner implementation. It can be added to the action using its ”body” property under the semantics tab of properties view. Different roles associated with the process model can be modelled using partitions in activity diagram, which is a similar concept as swimlanes in BPMN. We now present different view of this tool.

2.4.2 Main Verification view - Process tab

ProVer main view allows to launch verification of selected process model.

In every process we want to verify the completion i.e reach an ActivityFinal node in different mode i) `Universal`: all existing path reach an ActivityFinal node. ii) `Existential` It exist at least one path witch reach an ActivityFinal node.

Another important configuration is the BMC Limit. Effectively the formal method used to verify the process model is Bounded Model Checking (BMC) with Solver Modulo Theory (SMT) solver. This technique checks if all properties are satisfied until this bound. The only SMT solver available on this tool is z3 (MIT License, Microsoft Research) And the last configuration possible is dead transition that means a node is never reachable. We can see an example of configuration in Figure 26 Tool overview

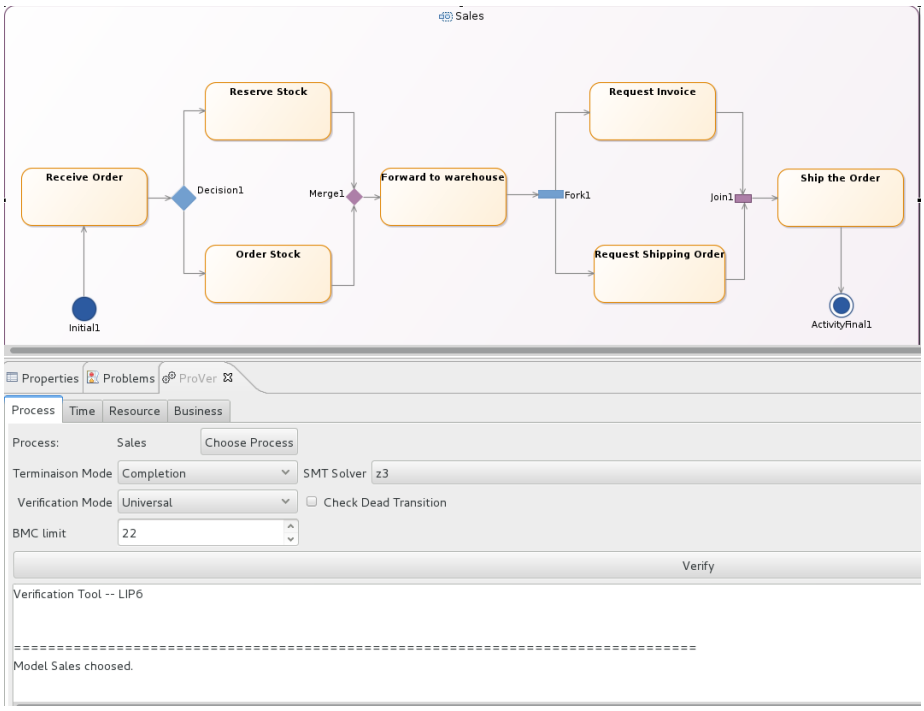

Figure 26 Tool overview

2.4.3 Time tab

Often, each activity is assigned with a duration and we have a fixed duration to finish the entire process model. ProVer allows checking this property automatically. We can affect duration to each action and a global time.

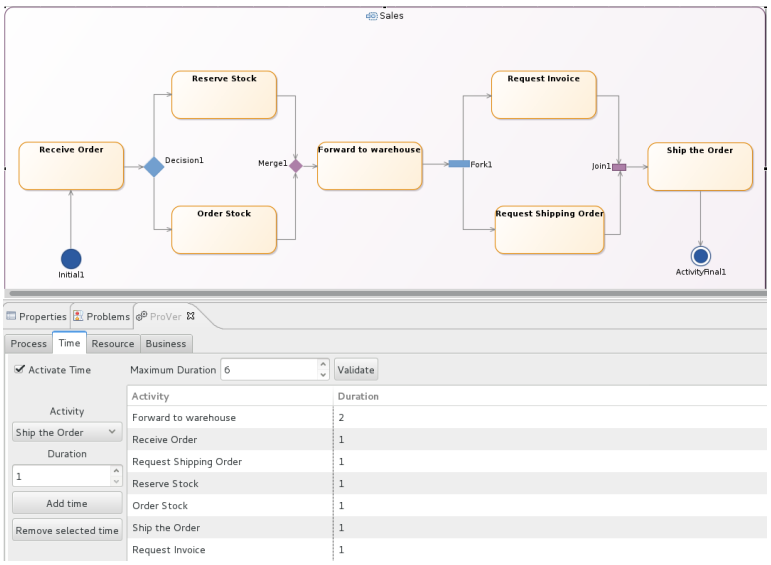

Figure 27 Time tab

As one can see in Figure 27 Time tab, we affect each activity one time unit expect `Forward to warehouse` activity witch assigned with two time units. The global time assigned to this process is 6 time units. Prover verifies the process can be reach an ActivityFinal node in "global time" with affected time units to each activities (`In Time` property)

Here detailed presentation of this view

- `Activate Time`: Enable or disable time constraint verification
- `Validate`: Validate maximum time to use during process
- `Add Time`:Add a new time constraint to a selected activity
- `Remove selected time`: Delete a time constraint for a selected activity

2.4.4 Resource Tab

This tab allows specifying resource constraints associated to process model activities. Such resource constraint are specified by a constraint name defining the resource class and an integer defining the maximum cardinality for instances of this class allocated to the constrained activity.

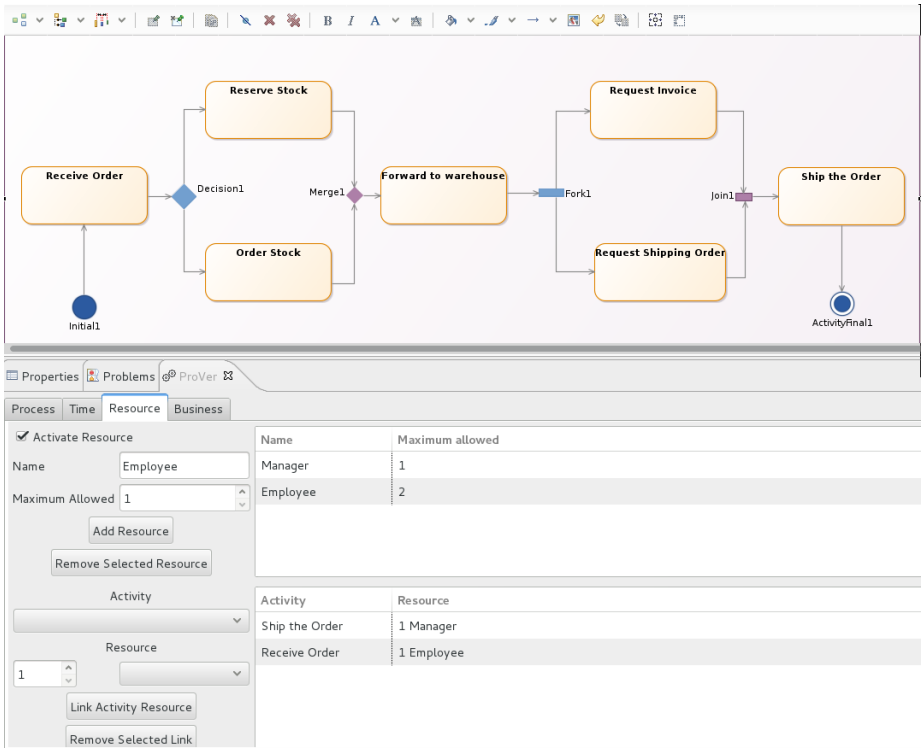

Figure 28 Resource tab

In Figure 28 Resource tab, we can see a configuration of resources (Manager and Employee) where there is at most 1 manager and 2 employees `Ship the order` was affected to manager and `Receive Order` was affected to 1 employee. Many resources can be assigned to each activity.

Here detailed presentation of this view

- `Activate Resource`: Enable or disable verification for the resource constraints specified
- `Add Resource`: Create a number of resource witch allow using on parallel
- `Remove selected resource`: Remove selected resource
- `Link Activity Resource`: Create a resource constraint by linking a created resource to a selected activity.
- `Remove selected link`:Remove a resource constraint by unlinking a created resource with an activity.

2.4.5 Business Tab

This tab allows specifying business constraints to be verified by the process model. Different kind of business properties exists:

- Relational: Before, After, in Parallel, in Exclusion
- Occurrence: Selected activity must be executed at least N times and/or at most M times before the process enactment terminates.
- Time usage: Selected Activity or Resource must be used before / after a time T

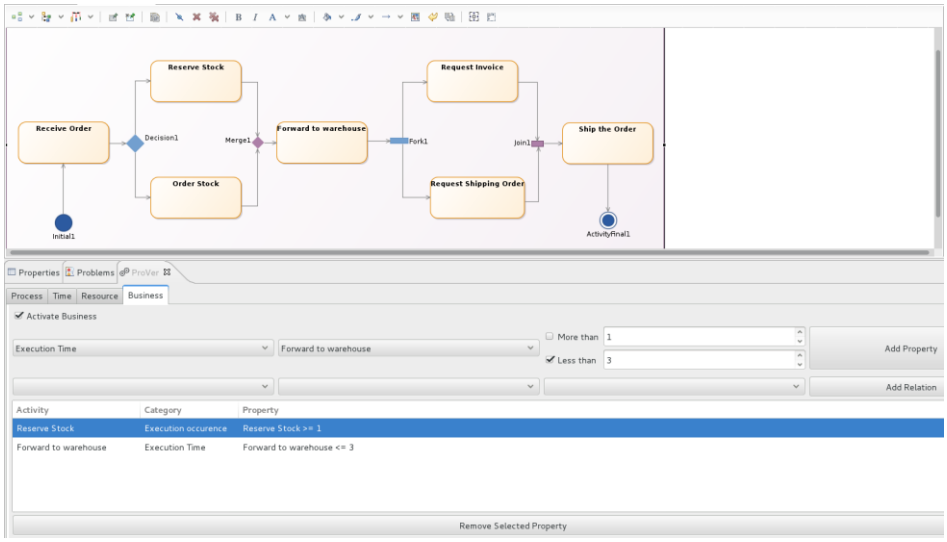

Figure 29: Business tab

We can see an example in Figure 29: Business tab, two business properties are added

- `Reserve Stock` must be executed at least one times, `Forward to warehouse` must be execute before global time inferior or equal to 3

Here detailed presentation of this view

- Activity execution occurrences:
- Activity execution time: selected activity A is executed at least Tmin and/or at most Tmax time units
- Resource usage time: selected resource is used at least Tmin and/or at most Tmax time units
- Data availability time: data on selected activity output pin P is available at least Tmin and/or at most Tmax time units.
- Relation Before: Selected activity A is executed before selected activity B
- Relation After: Selected activity A is executed after selected activity B
- Relation Parallel: Selected activities A and B must be executed in parallel
- Relation Exclusion: Selected activities A and B are not both executed during enactment. Either A is executed by B is not, or B is executed and A is not, or neither A nor B are executed.

2.5 Create Model

We now explain how to create a model.

1. Choose `File` → `New` → `Other...` This shall open a wizard
2. In this window, choose `UML Project` in `UML Designer` folder and click on `Next` (see Figure 30 Creation UML project)

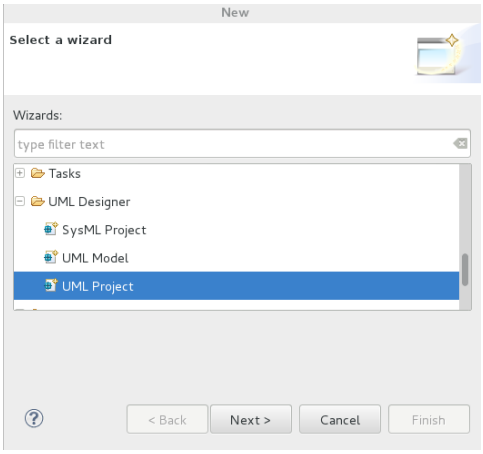

Figure 30 Creation UML project

3. Enter your project name and click on `Finish`
4. Eclipse proposes to open Modelling perspective, click on Yes. If you click on No you can open the perspective by clicking on at the top right of the `Open Perspective` Button, this shall open a window and choose `Modelling`. (Eclipse must be installed with modeling package in eclipse website).
5. On `Dashboard` click on on Activity Diagram
6. This shall open diagram window and you can then edit the activity diagram which properties you want to subsequently specify and verify using ProVer

2.5.1 Import Model

We can also import an exported model.

1. Go to `File` → `Import` This shall open a window
2. In this dialog box, choose `Existing Projects into Workspace` in `General` (see Figure 31 Import new project)
3. Browse the folder hierarchy to find the file to import, select it and then click on finish.

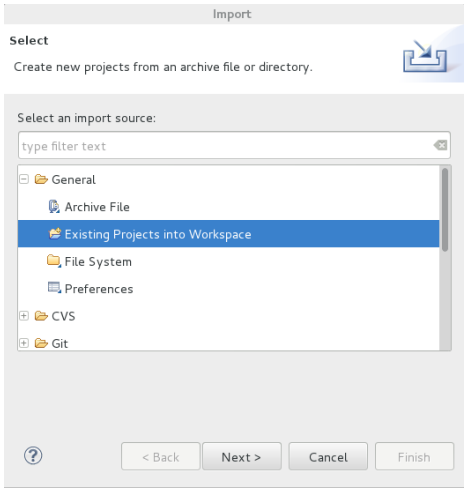

Figure 31 Import new project

2.6 Example

In this section we present step by step examples of process model verification using ProVer. In this first example we specify and then verify some properties which are `Completion` (mandatory) and `In Time` (defined by selected time units) on the Sales process available for importation from:(http://merge.lip6.fr/SalesExampleUML.zip) This activity is composed by 7 actions which describe a sales.

1. Open the diagram representation
2. Open the verification UI window by choosing `Window` → `Show View` → `Other`
3. Choose the ProVer option from Prover menu.
4. Click on on process, this shall write the name of the process on ProVer. (see Figure 32 Choose Diagram)

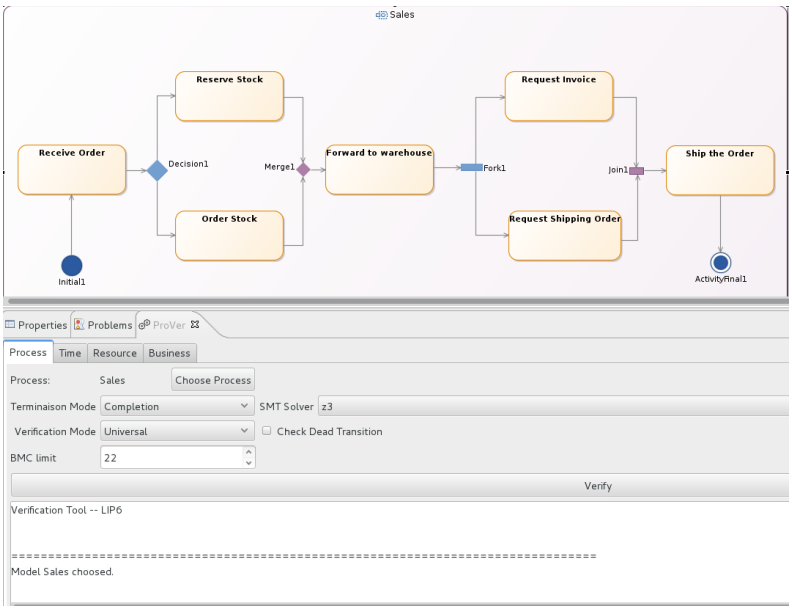

Figure 32 Choose Diagram

5. We can click on chosen process.
6. User now specify the process model properties which are `Completion` (mandatory) and `In Time` (defined by selected time units in time tab)
7. we can see in Figure 33 Time Properties all affected times and selected maximum duration of the process model.

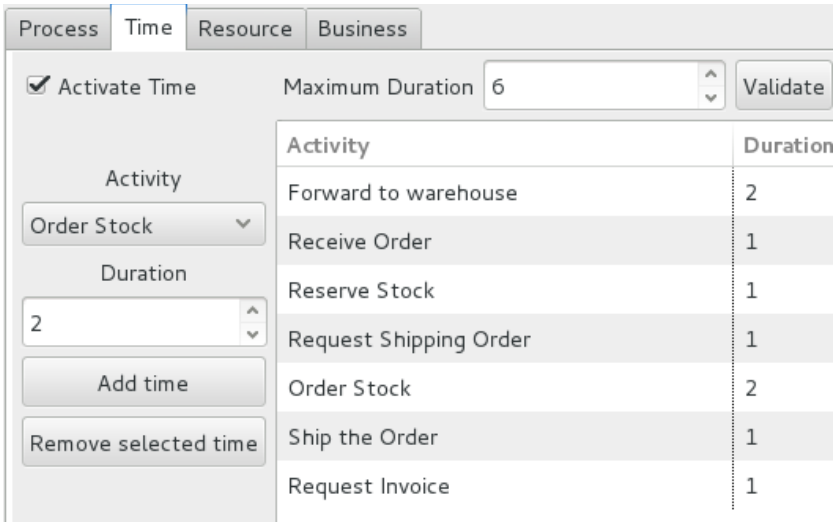

Figure 33 Time Properties

2.6.1 Result

This section present result of defined properties above. Verification with `universal` mode fails. The path violating global the time constraints. The root cause of the failure is also given in the verification log. It explains that the reason for the failure is: *Global time reach maximum value: 6. Need 7 time unit in total to complete time property* Prover highlights in red where the verification fails, in this example one more time unit is required to finish the process model.

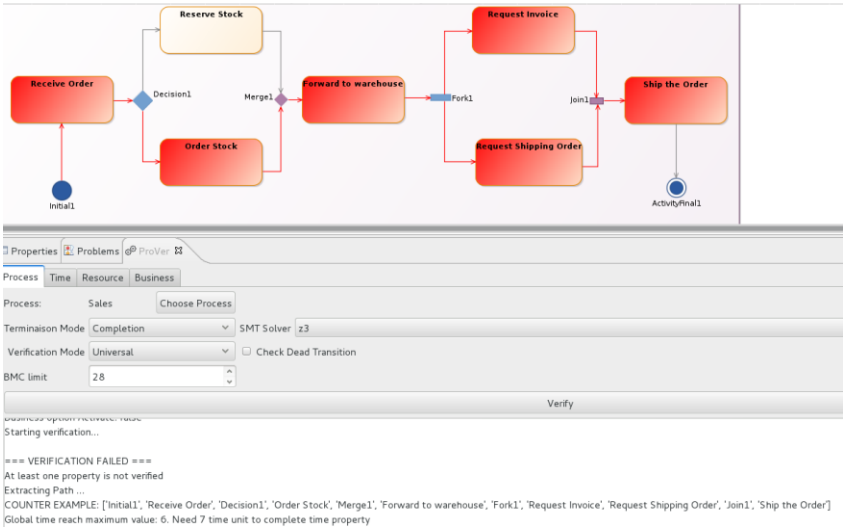

Figure 34 Verification Check Time Properties

But in `existential` mode verification succeeds. Effectively `Reserve Stock` activity is shorter than `Order Stock`, so it exist a path with 6 times unit. Prover highlights this path on green. (See **Figure 35 Verification Find Time Properties**)

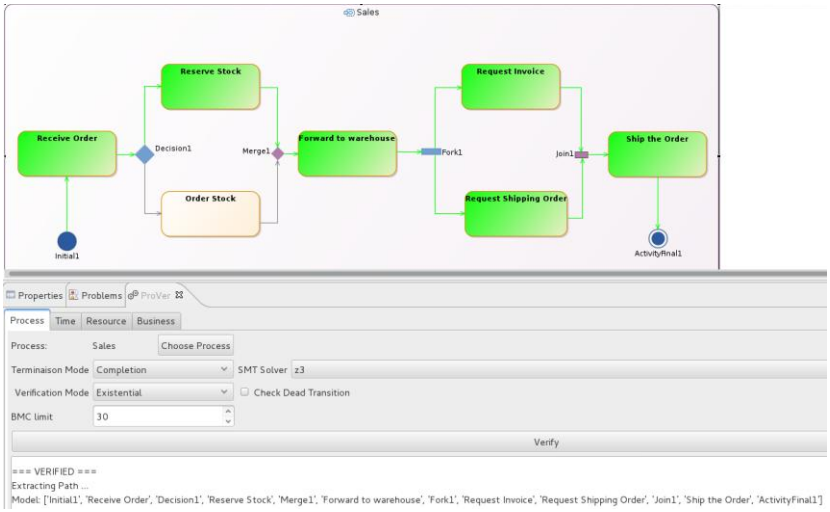

Figure 35 Verification Find Time Properties

2.7 Conclusion

Prover provides a set of properties that could be verified by anyone with no qualification on formal methods. The selection of all properties is entirely graphical and no special configuration is needed. All of these properties can be composed and verified all together. This tool is integrated in the MERgE platform and can be installed with an update site or locally with an archive file.

The use of ProVer provides an earlier detection of eventual flaws in process models, which avoids bugs, reduces the cost of production and a safer product for the end user.

The set of properties implanted in the tool can be customized to allow a more specific verification the process models but this requires some expertise on formal methods.

3 References

[Christov 14] S.C. Christov, G. S. Avrunin , L.A. Clarke, American Medical Informatics Association Annual Symposium (AMIA 2014), November 15-17, 2014, Washington, DC, pp. 395-404. (UM-CS-2014-022)

[Gruhn 07] V.Gruhn and R. Laue." What business process modelers can learn from programmers". Science of Computer Programming, 65(1) :4–13, 2007. 3

[Mendling 06]J an Mendling, M. Moser, G. Neumann, HMW Verbeek, B. F van Dongen, and W. MP van der Aalst. Faulty epcs in the sap reference model. In Business Process Management, pages 451–457. Springer, 2006. 3, 7

[Mendling 07] Jan Mendling, Gustaf Neumann, and Wil Van Der Aalst. Understanding the occurrence of errors in process models based on metrics. In On the Move to Meaningful Internet Systems 2007 : CoopIS, DOA, ODBASE, GADA, and IS, pages 113–130. Springer, 2007. 3

[Mendling 09] Jan Mendling. Empirical studies in process model verification. In Transactions on Petri Nets and Other Models of Concurrency II, pages 208–224. Springer, 2009. 3, 4

[Simidchieva 10]B I. Simidchieva, S. J. Engle, M. Clifford, A.C. Jones, S.Peisert, M. Bishop, L.A. Clarke, L. J. Osterweil, Proceedings of the 2010 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '10), August 9-10, 2010, Washington, DC. (UM-CS-2010-039)

[Vanhatalo 07] J.Vanhatalo, H. Völzer, and F.Leymann. Faster and more focused control-flow analysis for business process models through sese decomposition. In ICSOC 2007, pages 43–55. Springer, 2007. 3, 33, 143