Activity Diagram in the Unified Modeling Language (UML)

Course name: Method Engineering
Course code: INFOME

Faculty of Science
Department of Information and Computer Science
Utrecht University
Princetonplein 5,
3584 CC, Utrecht,
The Netherlands

Student Assistant: Daniel Alami

Student : Drilon Ferati- 5676932
Introduction

Mingsong Xiaokang and Xuandong (2006) define Unified Modeling Language (UML) as a standard visual modeling language which is designed for the purpose of specifying, visualizing, constructing and documenting the artefact of software system. UML was introduced in the mid-90s by Rumbaugh, Jacobson and Booch (2004) and it is nowadays acknowledged as the standardized language for modeling the structure and behavior of the system, with around 13 types of diagrams that help model the systems (Pilone & Pitman, 2005). The diagrams are separated in groups of structural modeling (that captures the static feature of the system) and behavioral modeling (that describes the interaction in the system).

Activity diagrams, as part of behavioral UML diagrams, are used to graphically represent workflows. They describe the sequential or concurrent flows of activities in the system (Mingsong et al., 2006). Activity diagram displays a sequence of activities, beginning from the starting point of the activity until the finishing point, by describing in detailed the decisions along the progress of the events in the activities and the overall flow of control. The simplicity of activity diagram and the ease of understanding, enables it to find applicability in a variety of modeling cases such as business process modeling (Brad, 2016), complex operation modeling, object flow modeling, and lately it is deliberately used for test case and use case generation. Although this technique does not have an explicit creator, the introduction of the UML language as a whole signifies also the creation and introduction of activity diagram.

For compiling an activity diagram a number of procedures should be covered both before and during the creation of the diagram. Although these procedures are not of a standardized nature, they are advocated as means that help in better activity diagram compilation. The following procedures are combined from the work of Gomaa, (2011), “Constructing Activity Diagrams,” (n.d.) and Embarcadero Technologies (2009):

- Identify the process
- Identify the activities
- Identify the actors
- Identify conditions
- Add swimlane
- Add transition
- Review the diagram

For constructing the activity flow a number of notations are used to describe each state of the activity flow such as the starting point, ending point, activities, decisions etc. The notations of Activity Diagram are very similar with the notations of statechart diagram and Petri nets (Eshuis & Wieringa, 2001). Table 1 represents the most used notations in constructing an activity diagram, which are adapted from Lucidchart, (n.d.), together with a detailed description for the corresponding notation. These notations are also explained in the work of (Eshuis & Wieringa, 2003).
<table>
<thead>
<tr>
<th><strong>Notation</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Start</strong></td>
<td>represents the beginning of a process or workflow in an activity diagram.</td>
</tr>
<tr>
<td><strong>Activity</strong></td>
<td>indicates the activities that make up a modeled process and it is the main component of an activity diagram.</td>
</tr>
<tr>
<td><strong>Connector</strong></td>
<td>arrowed lines that show the directional flow, or control flow, of the activity. An incoming arrow starts a step of an activity; once the step is completed, the flow continues with the outgoing arrow.</td>
</tr>
<tr>
<td><strong>Join (synchronization bar)</strong></td>
<td>it combines two concurrent activities and re-introduces them to a flow where only one activity occurs at a time and it can be represented vertically or horizontally.</td>
</tr>
<tr>
<td><strong>Fork</strong></td>
<td>splits a single activity flow into two concurrent activities.</td>
</tr>
<tr>
<td><strong>Decision</strong></td>
<td>represents the branching or merging of various flows with the symbol acting as a frame or container.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>allows the diagram creators or collaborators to communicate additional messages that don't fit within the diagram itself.</td>
</tr>
<tr>
<td><strong>Receive signal</strong></td>
<td>demonstrates the acceptance of an event. After the event is received, the flow that comes from this action is completed.</td>
</tr>
<tr>
<td><strong>Send signal</strong></td>
<td>means that a signal is being sent to a receiving activity, as seen above.</td>
</tr>
<tr>
<td><strong>Option loop</strong></td>
<td>used for modeling a repetitive sequence within the option loop symbol.</td>
</tr>
<tr>
<td><strong>Flow final</strong></td>
<td>shows the ending point of a process' flow.</td>
</tr>
<tr>
<td><strong>End</strong></td>
<td>represents the completion of a process or workflow.</td>
</tr>
</tbody>
</table>

*Table 1* - Notations of Activity Diagram
Example

Examples for creating activity diagrams can be identified easily on our daily life. One of those examples is shown in the following picture, which describes the workflow of the OV chip card scanners in the public transportation methods in Netherland.

**Picture 1 - Activity Diagram of OV chip card scanners**

As mentioned earlier, in Picture 1, we can see the workflow of the OV chip card scanners in the public transportation methods. The actors in this case is the user of the OV chip card who interacts with the card scanner. This indicates that the activity diagram describes the workflow of the card scanner system via a series of use cases with the first one being the activity of “Scan the OV chip card” in the scanners in order to check in. This is followed by a decision of the scanner whether the user has sufficient funds in their OV card or not. If the user does not have sufficient founds a set of concurrent activities that are represented by the fork follow, such as the “Illuminate red light” activity, that symbolizes the incomplete check in because of insufficient funds and “Notify that user has insufficient funds” appears on the screen of the scanner, from where the final activity of “Take away the card from the scanner” follows thus completing the activity flow.

If in the other case, the user does have sufficient funds in their card for the travel, a set of concurrent activities follow from the “fork”, such as “Illuminate green light” that indicates that the check in for the trip is done successfully, “Current balance on the card” which indicates how much funds are left in the card of the user for future travels and “Charged amount for the trip” which indicates how much is charged from the card of the user for the current trip. These concurrent activities are then synchronized via the join (synchronization bar), which introduces us to the last activity of “Take away the card from the scanner”, which is the same with the last activity in the case of not having sufficient funds.
Process Deliverable Diagram

The following segment introduces the process deliverable diagram of Activity Diagram. Referring to the work of Van de Weerd and Brinkkemper (2008) process deliverable diagram (PDD) is a meta modeling technique especially developed for method engineering purpose. Van de Weerd and Brinkkemper (2008) state that “PDD’s are used for analyzing, storing, selecting and assembling the method fragments” which is in line with the steps defined by van de Weerd, Brinkkemper, Souer, and Versendaal (2006) for situational method engineering.

In order to compile a PDD, Saeki (2003) approach was taken into consideration, where he proposed the diagram to have a meta process model, which is based on UML Activity Diagram in the left hand side, and a meta-data model, based on UML class diagram and depicted on the right hand side. The purpose of this diagram is to link the semantic information to the artefacts and in the same time to determine their quality in using this information.

Aside from the PDD an additional activity table is constructed, where all the activities and sub activities are listed together with a description about each of them, and a separate concept table, where all the concepts are listed together with a description about their meaning.

Activity Diagram is a modeling technique that finds applicability in a variety of modeling cases, whether that is for modeling business processes, generating use cases, test cases etc. Even though it is widely applicable, Activity Diagram up to date has no standardized procedures for its compilation. Most of the times the procedure that is followed to develop an Activity Diagram varies to the subject at hand. This making it difficult to produce a proper PDD for Activity Diagram. Figure 1 depicts a PDD of Activity Diagram where the activities represented are combined from a number of sources such as Gomaa (2011), “Constructing Activity Diagrams” (n.d.), Embarcadero Technologies (2009) and Omg (2010). Nevertheless this does not necessarily mean that these activities should be followed every time an Activity Diagram is modeled, as said earlier it all depends to the study at hand for which you are generating an Activity Diagram and thus in some cases some of the activities depicted in the figure below can be omitted (i.e. you can have an Activity Diagram without a swimlane).
### Activity Diagram Identification Process

<table>
<thead>
<tr>
<th>Activity</th>
<th>Sub activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification process</td>
<td>Identify the process</td>
<td>Identify what PROCESS you will model with the Activity Diagram. Whether it will be a business process, a use case, test case etc.</td>
</tr>
<tr>
<td></td>
<td>Identify activities</td>
<td>Identify the ACTIVITY of the PROCESS that you are modeling.</td>
</tr>
</tbody>
</table>
Identify actors
By identifying the ACTOR, it is determined who is responsible for each ACTIVITY.

Identify conditions
Identify all the CONDITION there are in the PROCESS that need to be satisfied in order to continue to the next ACTIVITY.

Development phase
Add the swinmlane to separate the activities of different actors
Add the SWIMLANE in order to specify the ACTIVITY that are conducted by a certain ACTOR.

Add transition among the activities
Add the TRANSITION to show the movement from one ACTIVITY to the other, and the direction of the movement.

Review the diagram
Review the DIAGRAM to see whether all the required ACTIVITY have been added and whether there are CONDITIONs that have not been covered.

<table>
<thead>
<tr>
<th>Table 2- Activity table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Concept</strong></td>
</tr>
<tr>
<td>DIAGRAM</td>
</tr>
<tr>
<td>PROCESS</td>
</tr>
<tr>
<td>ACTIVITY</td>
</tr>
<tr>
<td>LOCAL PRE-CONDITION</td>
</tr>
<tr>
<td>LOCAL POST-CONDITION</td>
</tr>
<tr>
<td>ACTOR</td>
</tr>
<tr>
<td>CONDITION</td>
</tr>
<tr>
<td>SWIMLANE</td>
</tr>
<tr>
<td>HIERARCHICAL PARTITION</td>
</tr>
<tr>
<td>TRANSITION</td>
</tr>
<tr>
<td>REVIEW</td>
</tr>
</tbody>
</table>

| Table 3- Concept table |
Related literature

A literature review on activity diagram indicated that this type of UML diagram found applicability in a wide number of researches for a variety of modeling cases. Dumas and Ter Hofstede (2001) examined how adequate is the activity diagram for workflow specification and at the same time evaluated its ability for capturing workflow patterns, whereas Kalnins and Vitolins (2006) in their work referred to activity diagram as one of the standards for design, execution and administration of business processes together with Business Process Language Notation (BPMN). Torres (2004) used activity diagram in their work for modeling agent (which are software entities designed to satisfy specific conditions such as goals) plans and actions. Other than the previously mentioned researches where activity diagram modeling finds applicability, an emerging domain that uses this kinds of diagrams in the latest years is the one of test case and use case generation using this type of diagram.

One of the papers that clearly explains why activity diagrams find applicability in test case generation is the paper of Kundu & Samanta, 2009. They state that the main reasons of using design model for program testing is that they can test the dynamic behavior of the system. They make it easier for software testers to better understand the system and find test information with simply comparing the models with the code, and the model based test case generation can be conducted at an early stage of software development cycle by allowing to carry out functions such as coding and testing in parallel. The results of the research indicated that this approach engendered in reduction of testing effort since it identifies the location of the fault in the implementation and at the same time this approach was capable of detecting faults in the loops and synchronization.

There are other authors that also used activity diagram for test case generation, as the ones mentioned in the deliverable example, other authors such as Mingsong et al. (2006) use activity diagram as design specification, in their work, and develop an automatic approach of test case generation, Chen, Mishra, and Kalita (2008) also propose an automated test case generation approach for the activity diagram since they see the lack of automated techniques for test generation as one of the bottlenecks of the UML activity diagram. Heinecke, Bruckmann, Griebe, and Gruhn (2010) propose an approach for generating acceptance tests of a high level automatically from business processes, which processes are modeled as UML activity diagrams.

The mentioned literature indicates that the activity diagram finds applicability in a wide range of domains, whether it is for modeling business processes, generating test case scenarios or generating use case scenarios.
References


**Key Terms**

**Method engineering**- the engineering discipline to design, construct and adopt methods (Brinkemper, 1996)

**Method fragment**- a coherent piece of an IS development method (Brinkemper, 1996)

**Situational method** – information system development method tuned to the situation of the project at hand (Harmsen, Brinkkemper, & Oei, 1994)