An Approach to Derive the Use Case Diagrams from an Event Table

Mohammad I. Muhairat and Rafa E. Al-Qutaish
Department of Software Engineering
Al-Zaytoonah University of Jordan
Airport Street, P.O. Box: 130, Amman 11733
JORDAN
mohmuhaba14@yahoo.com, rafa@ieee.org

Abstract:- Building the use-case diagram is a very important task since it represents a transition between the requirements and design phases. However, building such diagram is a time consuming process and needs a complete understanding of the requirements. In this paper, we introduce an approach to derive use case diagrams from an event table. This new approach will facilitate and speed the generation process of the use case diagrams. However, this approach will completely depends on the availability of a comprehensive event table which to be built from the available requirements.

Keywords:- Software Requirements, Software Design, Event Table, Use Case Diagram.

1 Introduction
Nowadays, there are many different approaches in the software engineering literature to identify the use cases, for examples:
1. Listing all users and define there needs [1], [2], [3], [4], [5];
2. Defining all system functions and adding new functions that user may be need [1], [2], [3], [4], [5];
3. List all graphical user interfaces [2],[6];
4. Defining all users' goals of system [1], [2], [7], [8], [9].

In industry, many practitioners are using the third approach to get an initial list of use cases. Furthermore, the event decomposition technique [1], [2], [5] is the most used one for defining a use case model. This technique focusing on the events a system must respond to and looking at how a system responds.

An event is an occurrence at specific time and place, can be described, and should be remembered by the system [2], [11] [16].

Building the use-case diagram is a very important task since it represents a transition between the requirements and design phases. However, building such diagram is a time consuming process and needs a complete understanding of the requirements. In this paper, we introduce an approach to derive the use case diagrams from an event table. This new approach will facilitate and speed the generation process of the use case diagrams. However, this approach will completely depends on the availability of a comprehensive event table which to be built from the available requirements.

The rest of this paper is organized as follows: Section 2 presents a general overview on the related concepts, that is, on the event table and use case concepts. In Section 3, the approach of deriving the use case diagram from the event table has been explained. Section 4 gives an example of the implementation of the proposed approach. Finally, Section 5 concludes the paper and introduces the potential future works.

2 Related Concepts

2.1 Event Table
Since 1980's, the event analysis technique [10] [11] [12] has been the preferred one for the practitioners. The results of event analysis are documented in an event table. In the structured approach, event analysis recognizes a basic set of processes. Whereas, in the object-oriented analysis, each event discourse an essential use case [13]. Furthermore, an event table can be created from the external events to support the use case diagrams [14]. In addition, the event table has been used by Gargantini and Heitmeyer [15] to generate a suite test sequences.

In the event table, there are three types of events, that is:
1. External Event (EE): an event that occurs outside of the system, usually initiated by external actor or user; for example: student wants to search for a book item.
2. **Temporal Internal Event (TIE):** an event that happens when the system reaches a specific point of time; for example: *time to print book items report.*

3. **Conditional Internal Event (CIE):** an event occurs when something happens inside the system and the system must initiate some process to response for this event; for example: *student reordered new books when the reordered point is reached.*

Business modeling help analyst to understand the business process. As result of that modeling, business events are identified and documented in an event table. Event table is a list of actions that lists events in rows and the information about each event in columns. Analyst can use event table to define use case model. However, analyst has to make some decisions when building a use case model.

The first one is to combine business events into one use case. For example, the business events of adding, deleting, updating customer information, analyst can combined them in one use case-Maintain Customer Information. Also, analyst makes decisions to split one business event into multiple use cases. For example, the business event customer withdraws his cash, analyst can split it two use cases Withdraw Cash and Identify Customer and can identify the relationship between them, for example: *Withdraw Cash <<include>> Identify Customer.*

From the above example (*student wants to search for a book item*) and for an EE, we can split this event into three main parts:

\[
\text{EE} = \begin{cases} 
  \text{Student - Source} \\
  \text{Search - Action} \\
  \text{Book item-affected object}
\end{cases}
\]

In addition, the TIE example (*time to print book items report*) can be divided into three main parts as the following:

\[
\text{TIE} = \begin{cases} 
  \text{Time or "System" - Source} \\
  \text{Print - Action} \\
  \text{Book item - Affected object}
\end{cases}
\]

Finally, we can split the CIE (*student reordered new books when the reordered point is reached*) into the following three main parts:

\[
\text{CIE} = \begin{cases} 
  \text{Student - Source} \\
  \text{Reordered - Action "if the reordered point is reached"} \\
  \text{Book item - Affected object}
\end{cases}
\]

As a result of this splitting, analyst defined the core elements "columns" for the proposed event table as in Table 1.

**Table 1: Event Table.**

<table>
<thead>
<tr>
<th>Event</th>
<th>Source</th>
<th>Action</th>
<th>Object</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where:

- **Event:** an event which causes the system to do something.
- **Source:** the source of an event (an actor for an EE and the system for a TIE and CIE).
- **Action:** the system functionality which we need.
- **Object:** The object affected by this action.
- **Destination:** An actor that receive the result of an event execution

For the purpose of building a complete use case model, we can extend this table and make some modification to contain other elements, see Table 2.

**Table 2: Extended Event Table.**

<table>
<thead>
<tr>
<th>Event</th>
<th>General Source</th>
<th>Special Source</th>
<th>Action</th>
<th>Object</th>
<th>Includes &quot;Action&quot;</th>
<th>Extends &quot;Action&quot;</th>
<th>Specializes &quot;Action&quot;</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Where:

- **General Source or Special Source:** The type of an event source. We used these columns to define a Generalization / Specialization relationship between sources.

- **Includes:** We used this column to determine the existence of includes relationship between actions.

- **Extends:** We used this column to determine the existence of extends relationship between actions.
Specializes: We used this column to determine the existence of specializes relationship between actions.

2.2 Use Case Diagram
Use case diagram captures the system or subsystem behaviour. It represents the interaction between the actors and pieces of functionality called use cases. An actor is an idealization of a role played by an external person, process, or thing interacting with the system, subsystem, or class [7], [18]. Actors participate with one or more use cases by exchanging messages.

Actors may be defined in generalization hierarchies, in which an abstract actor description is shared and augmented by one or more specific actor descriptions. An actor may be a human, a computer system, or some executable process. An actor is drawn as a small stick person with the name below it. A use case is a coherent unit of externally visible functionality provided by a classifier (called the subject) and expressed by sequences of messages exchanged by the subject and one or more actors of the system unit [17], [18]. A use case can participate in several relationships; in addition to association with actors (see Table 3).

Recently, Snoeck [16] has proposed a new form of event table which is called object-event table to be a useful technique for modelling interaction between domain object types.

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Function</th>
<th>Notation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Association</td>
<td>To indicate the communication between actors and uses cases.</td>
<td></td>
</tr>
<tr>
<td>Extend</td>
<td>To indicate the insertion of additional behavior into a base use case.</td>
<td>&lt;&lt; extend&gt;&gt;</td>
</tr>
<tr>
<td>Include</td>
<td>To describes a behavior that is inserted explicitly into a base use case.</td>
<td>&lt;&lt; include&gt;&gt;</td>
</tr>
<tr>
<td>Use case or actor generalization</td>
<td>To indicate the communication between a general use case (actor) and a more specific use case (actor) that inherits and adds features to it.</td>
<td></td>
</tr>
</tbody>
</table>

3 Deriving the Use Case Diagram
This process consists of six steps, to build and check the correctness of the use case diagram.
1. Identify the actors for each event or action from the sources and destinations.
2. Identify the relationships between actors, if exists. There is only one type of relationship between actors, that is, a generalization / specialization relationship.
3. Identify the use cases. The analyst can derive the use case from the action which proceeded by an actor.
4. Identify the relationships between use cases, if exists. As we mentioned above, there is three type of relationships between use cases (includes, extends, and specializes).
5. Integrate all use cases and actors with all relationships types in one use case diagram.
6. Test the use case diagram.

From a given event table we can map the use cases and the actors, as in Figure 1.

![Figure 1: Mapping the Use Cases and Actors from an Event Table.](image-url)
However, this mapping could be implemented using the two types of events, that is, external event (EE), and internal events (TIE, CIE). Figures 2 and 3 illustrate examples of both types, respectively.

<table>
<thead>
<tr>
<th>Event</th>
<th>Source</th>
<th>Action</th>
<th>Object</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student wants to search for a book item</td>
<td>Student</td>
<td>Search for a book item</td>
<td>Book item</td>
<td>Student</td>
</tr>
</tbody>
</table>

**Figure 2:** An Example of Mapping a Use Case and Actor from an EE.

<table>
<thead>
<tr>
<th>Event</th>
<th>Source</th>
<th>Action</th>
<th>Object</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to print a book items report</td>
<td>System</td>
<td>print a book items report</td>
<td>book items</td>
<td>Library Employee</td>
</tr>
</tbody>
</table>

**Figure 3:** An Example of Mapping a Use Case and Actor from a TIE and CIE.

5 An Example
To demonstrate the feasibility of our approach, this approach has been applied for a case study. This section will implement the process of deriving a use case model from an event table for car rental web site, see Table 4 for its event table.

<table>
<thead>
<tr>
<th>Event</th>
<th>General Source</th>
<th>Special Source</th>
<th>Action</th>
<th>Object</th>
<th>Includes &quot;Action&quot;</th>
<th>Extends &quot;Action&quot;</th>
<th>Specializes &quot;Action&quot;</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer browses the car models index</td>
<td>Customer</td>
<td>Member</td>
<td>Browse the car models index</td>
<td>Car model</td>
<td>View results</td>
<td>-</td>
<td>Look for a car model</td>
<td>Customer</td>
</tr>
<tr>
<td>Customer searches for a car model</td>
<td>Customer</td>
<td>Member</td>
<td>searches for a car model</td>
<td>Car model</td>
<td>View results</td>
<td>-</td>
<td>Look for a car model</td>
<td>Customer</td>
</tr>
<tr>
<td>Customer views the results</td>
<td>Customer</td>
<td>Member</td>
<td>view results</td>
<td>Car model</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Customer</td>
</tr>
<tr>
<td>member log on to the web site</td>
<td>Member</td>
<td>-</td>
<td>Log on to the web site</td>
<td>Web page</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Member</td>
</tr>
<tr>
<td>member view his details</td>
<td>Member</td>
<td>-</td>
<td>View member details</td>
<td>Member</td>
<td>-</td>
<td>member log on to the web site</td>
<td>-</td>
<td>Member</td>
</tr>
<tr>
<td>member log off from the web site</td>
<td>Member</td>
<td>-</td>
<td>Log off from the web site</td>
<td>Web page</td>
<td>-</td>
<td>member log on to the web site</td>
<td>-</td>
<td>Member</td>
</tr>
<tr>
<td>Member makes a reservation</td>
<td>Member</td>
<td>-</td>
<td>Make reservation</td>
<td>Reservation</td>
<td>-</td>
<td>member log on to the web site</td>
<td>-</td>
<td>Member</td>
</tr>
<tr>
<td>Member views a reservation</td>
<td>Member</td>
<td>-</td>
<td>view reservation</td>
<td>Reservation</td>
<td>-</td>
<td>member log on to the web site</td>
<td>-</td>
<td>Member</td>
</tr>
</tbody>
</table>
The following are the steps which will be used to build the use case diagram from the given event table (Table 4) for our example:

1. From the sources and destinations fields, we can define three actors: customer, member and non-member.
2. From the general and special sources fields, we can define a generalization / specialization relationship between actors.
3. From the actions field, we can derive all the use cases, as the following:
   - Browse a car models index,
   - Searches for a car models,
   - View results,
   - Logs on the web page,
   - View member details, and
   - Logs off the web page.
4. From the "includes action", extends action, and specialize action fields, we can define all the relationship types between the use cases to be includes, extends and specialize. In addition, a new use case can be defined from specializes action field.
5. Now, we will integrate the all elements to build the use case diagram, as in Figure 4.

![Use Case Diagram](image)

**Figure 4:** The Integrated Use Case Model.

6. Finally, the following steps could be used to test the use case diagram and to test the use cases against functional requirements:
   a. Check if all events are covered by actions "use cases;"
   b. Check if each use case represents a functional requirement;
   c. Check if all actors are well defined;
   d. Check if the relationships between the actors are defined;
c. Check if the relationships between the use cases are defined; and
f. Check the relationship between the different actors and use cases.

6 Conclusions and Future Work
Constructing the use-case diagram is a very important and essential task to go ahead to the design process. However, the use case diagram represents a transition stage between the requirements and design phases. Furthermore, building such a diagram is a time-consuming task and needs a complete understanding of the user requirements. In this paper, we have introduced an approach to derive the use case diagram from an event table. This new approach will facilitate and speed up the generation process of the use case diagrams. However, this approach is completely depending on the availability of a comprehensive event table which should be built during earlier tasks from the available user requirements.

It can be clearly noted from the above sections that this approach gives an ideal and reasonable methodology to build the intended use case diagram from any comprehensive event table. In addition, this approach will save the time for the building process of the use case diagrams.

As a future work, this approach could be automated by developing a CASE tool to take the event table as an input and produce the intended use case diagram as an output.

References