International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 3, Issue 4, July-August 2014 **ISSN 2278-6856**

An Alternate Approach for OO Analysis using Event Taxonomy and Templates

Jay Nanavati[1], Dr. Yogesh Ghodasara[2]

[1] School of Science, RK University, Rajkot, India, [1] SEMCOM, Vallabh Vidyanagar, India.
[2]College of Agricultural I.T., Anand Agriculture University, Anand, India

Abstract-*This paper discusses a systematic approach to classify and document events that take place during life-cycle of a system. This paper also proposes a changeover (mapping) process which uses Event Templates as input, apply one or more mapping rules to them and generate important information like Class name, Message passing(i.e. flow of information) among objects, Operations to be defined in classes, relationship among classes etc.*

Keywords-Events, Event taxonomy, Event-template, OOA

Introduction

A very common technique to identify objects in object-oriented modeling is to the requirement specifications for nouns that represent entities or objects. The basic problem with this approach is large number of nouns.Use Case Modeling technique proposed by Jacobson [1], is one solution to this problem. Instead of looking for nouns, the approach first breaksdown the entire scope of system functionality into a number of use cases. A use case is asequence of behaviorally related events that flow through a system [1, 2].

Relationship between Events and Objects

A typical object-oriented system can be viewed as a collection of objectsand their interaction with one-another.Objects interact and work together through events to offer overall functionality of the system.At any time, each object will be in a 'state'. This state is mainly represented by value of attributes at that time. If value of one or more attributes changes, the state of the objectalso changes.Objects also participate in relationship with other objects. For example, there may be aggregation or composition kind of relationship.Objects are the most active entity in the entire system.In the proposed approach, an object which initiates an event will be known as Initiator, the one which is affected by events, will be known as Affector. Those objects which support occurrence of events will be known as Facilitator.There will always be a Response associated with an event. The response will be either an activity or an action. Activity takes some time to complete whereas action completes instantly. In fact an activity may consist of one or more operations.

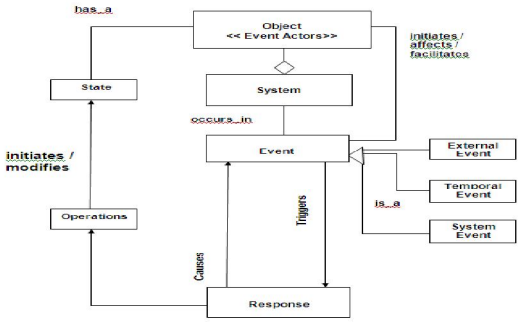

Figure 1 –Event-Object Interaction

An event can trigger a response and a response,Iturn,cangenerate new events. Thus, Events and Response have a cyclic relationship. [3]Further, responses and objects are indirectly related with each other. Response generates some information which may affect the state of objects. Objects generate events which trigger response. [4]

Representation of Events

There may be a large number of events occurring in a system. So each event must be identified uniquely.[5]A unique identifier (ID), name and description will be used to represent an event.Further, all the important information associated with each event must be recorded in a uniform manner. We propose a data structure, called an event template, to record all important information associated with events occurring within a system.

Event Template

In the beginning, events are extracted from requirements. Then one event template is created for each event.

The important fields of the event template are as follows:

- ✓ Event ID
- ✓ Event Name
- ✓ Event Description
- ✓ Role of Object
- ✓ Causative Event
- ✓ Trigger Event Vector
- ✓ State Changes
- ✓ Timestamp
- ✓ Input
- ✓ Output

The following table describes each of theseattributes further:

Sr. No.	Attribute	Type	Description
1	Event ID	String	A value to identify an event uniquely.
2	Event Name	String	A simple name expressed in natural language in Verb-Object form.
3	Event Description	String	Short description of an event in Subject-Verb-Object form.
4	Role of Object	String	Initiator/Facilitator/Affecter
5	Causative Event	Event ID	Event responsible for triggering given event.
6	Trigger Event Vector	Event IDs	A set of events which are triggered by given event.
7	State Changes	String	Changes caused by the event.
8	Timestamp	Time	Time at which the event occurs. Since it is not possible to predict the exact time, at analysis phase dummy timestamp can be used,
9	Input	Data	Input refers to the data needed by the event to trigger.
10	Output	Data	Output refers to the data generated by the event.
11	Count	integer	No. of instances participating in an event.

Use of Event Template

The proposed Event Template will be used to systematically and consistently (i.e. uniformly) record the event-related information. Such information then can be used to build an Event-based framework

Event template vs. Use case

Sr. No.	Area	Observations	
		Use Case	Event
1	Name	Use case name describes goal state to achieve.	Event name describes interaction of the object with the system.
2	Description	A sequence of related events.	Represents a single interaction.
3	Actors/Roles	Two categories: primary and secondary	Three categories: Initiator, Facilitator and Affecter
Sr. No.	Area	Observations	
		Use Case	Event
4	Timestamp	Not so important, used in sequence diagram.	Important, used in identifying temporal (timing-based) relationship among events.
5	Trigger	An event that initiates a use case.	Set of events responsible for causing an event.
6	Pre-conditions	Present	Causative events
7	Post-conditions	Present	Trigger Event Vector
8	Trigger Event Vector	Not Present	Present
9	Input,Output	Present	Not Present
10	State Changes	Not Present	Changes caused by events in the system state.
11	Normal Flow / Alternative Flow	Present	Determined by timestamps and trigger event vector.

Event Taxonomy (Classification)

There exists different ways to classify events. Each of them has been outlined following:

Internal event/ External event/Temporal event/ System event

An event that takes place as a result of an operation within the system is known as an Internal event.

Example: In computerized banking system, update of balance as a result of credit/debit operation.An event that takes place as a result of an operation outside the system is known as an External event.

Example: In PF management system, the fund is transferred when a member changes the job. An event that takes place as a result of some deadline or expiry of timer is known as a Temporal event.

Example: In case of postpaid mobile connection, monthly bill is generated and emailed to the customer.An event which occurs in response to a critical situation during system's working is known as a System event.

Example: In nuclear system, the system is shutdown if the radiation becomes dangerous.

Expected event/ Unexpected Event An event which is expected to occur as a result of some operation is known as an Expected event.

Example: In PF management system, when a member retires, it is expected that the withdrawal of fund will take place.An event for which it is not possible to predict if it will occur or not and when it will occur if it occurs at all, is known as an Unexpected event.

Example: In insurance system, it is difficult to predict if the claim will be made by insured person or when it will be made.Interestingly, temporal events are always expected event because they certainly occur after particular deadline or time period. Expected event can be used to determine action to be taken to deal with some predictable situation

Flow-oriented event/ Temporal event/Control event

An event which occurs as a result of data-flow in the system is known as Flow-oriented event.

Example: In online examination system, the result is prepared after the candidate clicks on the Submit button (i.e. when all the answers from the candidate are received).

An event that takes place as a result of some deadline or expiry of timer is known as a Temporal event.

Example: In case of library system, penalty is calculated if the book is not returned by the stipulated deadline.An event that occurs in response to some other event is known as a Control event.

Example: In online shopping, if the user confirms the order and makes the payment, an invoice is to be prepared.

Extraction of Events from Requirements

The proposed approach starts with extraction of events from requirements. Here the focus is on extracting such eventsthatrepresent a business operation which is performed by one person at one place.It should be noted that the extraction of events may be manual in case the requirement specification is of moderate size. However, most of the production-quality software projects involve a large requirements specification. Therefore, an NLP-based tool may be used. [6].Elementary events add a measurable value to the business.As a result of elementary event, the system obtains a consistent state. [7].Information needed for generating class diagrams is derived from the set of elementary events.

Step-by-step Process

Step-1 Extract Elementary events from textual requirements and generate a list of such events.

Input: Requirements expressed in written form

Output: Events List

Step-2 Try to explore the requirements more and search for similar events once more, to ensure that no event is missed out.

Input: Requirements expressed in written form

Output: Enhanced Events List

Step-3 Create an Event Template for each event present in the Events List.

Input: Events List

Output: Event Template for each event

Step-4 Apply the (proposed) Changeover process on Event templates to extract information of classes, their attributes, methods and inter-relationships.
Input: Event Template for each event
Output: Classes with attributes, methods and relationship with other classes.

The (proposed) Changeover Process

The Changeover process will be basically a mapping procedure. It will take Event Templates as input, apply one or more mapping rules to them and generate important information like

- ✓ Class name
- ✓ Message passing(i.e. flow of information) among objects
- ✓ Operations to be defined in classes
- ✓ Relationship among classes

Mapping rules to be used as a part of this process have been described next.

IFA-to-Class

Each of the Initiator/ Facilitator/Affector will be mapped to a class. There is a field in Event Template named "Role of Object". Value of this field will be extracted to get classname.Possible problem: Redundancy in classname may be there because more than one event may suggest the same class. Proposed Solution: Redundancy can be removed primarily by merging all classnames and then removing duplicates. Further, synonyms may also be removed.

IFA-to-Role

The same "Role of Object" field, discussed earlier, will help to add and set similar "Role" attribute in a class.

IFA-to-Classtype

This mapping will add and set a "Classtype" field in a class. It can take one of the following the values:

1) Entity class: An Entity class represents the entity in the problem domain. Information related to such entity has to be stored for long time.
2) Control class: Control classes are used to represent transaction, co-ordination, sequencing and controlling information.
3) Boundary class: Boundary classes define scope of the problem. They specify entities internal and external to the problem domain.

Initiator and Affecter can be Entity or Control class. Facilitator acts as Boundary class**.**

Count-to-Association

The "Count" field of an Event Template represents no. of instances participating in an event. It gives relationship of one with other. (One-to-one or one-to-many)

State Changes-to-State

State Changes section of an Event template describes type of a change that can occur due to an event; and corresponding input and output section describes attributes getting affected by events for carrying out such a change

If a State Changestype is creation (e.g. tour created), inputs from input section define new attributes (state members) of an Affecter class**.**

If a State Changes type is termination (e.g. tour terminated), an object already exist, so inputs are attributes of a specific affecter (object) to be terminated from a system.

If a State Changes type is read or access, an object already exist, so inputs search a specific object to be read or accessed. Such a change-event has a facilitator but no affecter.

If a State Changes type is modification/ updating (e.g. tour modified), an object already exist, so inputs are attributes to be modified/updated or some new attributes to be added to an affecter (object).

Calculate, Compute or Monitor are special cases of modify, read or access**.**

If a State Changes type is calculation, an object already exist, so inputs are attributes to be used to perform some calculations and modify an object state. Output of a change event produces important result in a system.

If a State Changes type is monitoring, an object already exist, so inputs are attributes to be checked to detect conditions for triggering a state or a control oriented event. Input is indicative of attribute(s) to be monitored. Output of a change event produces important result in a system.

If a State Changes type is computation, an object already exist, so inputs are attributes used to perform a query or compute a functional value without modifying an object state. Output of a change event produces an important result in a system.

State Changes-to-Association

An Initiator starts an event and an affecter gets affected by an event, so a direct relationship is mapped between them. If an event has a facilitator, an Initiator carries out an event with the help of a facilitator, so a relationship also exists between them.Whenever an event causes a change, such that the type is connection, then an association is mapped between an Initiator and an Affecter or an Initiator and a Facilitator of an event. This mapping rule affects a class diagram. When the type is disconnection, an object diagram is affected. A verb phrase from an event name is mapped to define an association name property of an association. A count attribute specifies multiplicity of an association. An association is automatically mapped by rules of creation, access, read, modify and classify.

State Changes-to-Access

Selector functions are defined in every class to read state of an object. Whenever an event causes a change, such that the State Changestype is read or access, then an association relationship is mapped between an Initiator and a Facilitator of an event, if not already mapped by an earlier rule. This mapping rule does not change state of an object. The read event is to read an entire state of an object whereas an access event only reads a part of an object's state. This rule adds a selector (an overloaded get method) to a facilitator (if it is an entity class) and correspondingly, adds a read or an access method to an initiator or a facilitator (if it is a boundary or a control class) of an event. A read method is given a name- read followed by name of a facilitator and an access method is given a name- access followed by name of an attribute of a facilitator.

State Changes -to-Modifier

Events modify (update) state of an object through modifiers defined in a class. Whenever an event causes a change, such that the State Changestype is 'update', then an association relationship is mapped between an Initiator and an Affecter or an Initiator and a Facilitator of an event. An update event only updates a part of an object state. This rule adds a modifier (i.e. an overloaded set method) to a facilitator (if it is an entity class) or an affecter and correspondingly, adds an update method to an initiator or a facilitator (if it is a boundary or a control class) of an event. Such a method is given a name- update followed by name of an affecter.

State Changes-to-Classification

If the State Changestype is denoted by word 'classified' such as A classified B, class A is

classified to be of type class B. Similarly words like 'type of', 'can be', 'is a', 'kind of' among Initiators, Facilitators or Affecters of events are mapped to inheritance.

Implementation and Practical Evaluation of the proposed approach

The proposed approach has been implemented on an imaginary Reservation System.

Requirements

Software for a travel agency provides reservation facilities for the people who wish to travel on tours by accessing a built-in network at the agency bureau. The application software keeps information on tours. Users can access the system to make a reservation on a tour and to view the information about the tours available without having to go through the trouble of asking the employees at the agency. The third option is to cancel a reservation that a user has made. Any complaints or suggestions that a client may have could be sent by email to the agency or stored in the complaint database. Finally, the employees of the corresponding agency can use the application to administrate the system's operations. Employees can add, delete or update the information on the customers and the tours. For security purposes, each employee is provided a login ID and password by the manager to be able to access the database of the travel agency.

Sr. No.	Event	Type of Event
1.	Customer views tour information.	External Event
2.	Customer makes a reservation on tour.	External Event
3.	Customer cancels a reservation on tour.	External Event
4.	Customer sends a complaint.	External Event
5.	Customer sends a suggestion	External Event
6.	TA Software provides user_id and password to customer.	External Event
7.	TA Software sends complaint to Travel agency.	Temporal Event
8.	TA Software sends suggestion to Travel agency.	External Event
9.	Manager provides login_id and password to employee.	External Event
10.	Employee adds customer information.	External Event
11.	Employees add tour information.	External Event
12.	Employees update customer information.	External Event
13.	Employees delete customer information.	External Event
14.	Employees update tour information.	External Event
15.	Employees delete tour information.	External Event
16.	Customer registers with TA software.	External Event
17.	TA software sends a complaint form to the Customer.	Temporal Event
18.	TA software sends a suggestion form to the Customer.	Temporal Event
19.	TA software generates a monthly report of all potential customers.	Temporal Event
20.	TA software monthly sends list of all tours to customers.	Temporal Event
21.	TA software weekly generates a report of all booked tours.	Temporal Event
22.	TA software weekly generates a report of all canceled tours.	Temporal Event
23.	TA software displays the tour details.	Temporal Event
24.	TA software generates monthly reports on the revenue.	Temporal Event

As per step 4 in the process, all events are documented in the proposed Event template.Event Templates corresponding to some of the events listed above are shown below:

- Event template for event "Travel agency enters tour information through TA software"

1.	Event ID	EA03
2.	Event name	Tour information
3.	Description	Travel agency keeps tour information through this application software
4.	Role of Object	Initiator
5.	Causative Events	EA01
6.	Trigger Vector	NULL
7.	State Changes	Connection event between Travel agency and tour creation event of Tour class
8.	Timestamp	TA2
9.	Input	Tour id, Tour name, Source, Destination, Cost, Days, Availability status
10	Output	Tour Information
11	Count	1

- Event template for event "Customer registers with TA software"

1.	Event ID	EA04
2.	Event name	Register customer
3.	Description	Customer registers with software
4.	Role of Object	Initiator
5.	Causative Events	Customer makes a reservation on tour.
6.	Trigger Vector	TA Software provides user_id and password to customer.
7.	State Changes	Creation event of Customer profile
8.	Timestamp	TA3
9.	Inputs	Customer name, email, phone number etc.
10	Output	Travel agency provides user_id and password to customer
11	Count	1

Application of (proposed) Changeover Process

The proposed rules were applied on all event templates of events listed above and information to generate a class diagram is extracted from these event templates. [8]

IFA-to-Class

In the case study chosen, we have identified Initiators, Facilitators and Affecters from all event templates and found the following potential class names:Customer, Tour, Reservation, Complaint, Suggestion, Manager, Employee

IFA-to-Role

Role of an Initiator, a Facilitator and an Affecter identified from the case study are shown below:

Customer – Initiator/Affector

Tour – Facilitator/Affector

Reservation - Facilitator/Affector

Complaint, Suggestion – Facilitator or Affector

Manager – Initiator

Employee – Initiator/Affector

IFA-to-Classtype

Entities have been classified as follows:

Customer, Tour, Reservation, Complaint, Suggestion, Employee – Entity class

Manager – Boundary class

Count-to-Association

In an event, "Customer makes at most four reservations for tour", multiplicity of a customer (initiator) is not specified so it is assumed to be default '0..1' and for a reservation (affecter) multiplicity count is at most four. The multiplicity constraint is specified with a class association

relationship with other classes and is not a property of a class alone.

State Changes -to-State

In an event, "TA Software registers a Customer" State Changestype is creation of a customer, so inputs like customer name, id, email, phone number and password, extracted from an event template of this event, define attribute list of a Customer (affecter class). In an event, "TA Software updates availability status of the tour booked", State Changestype is modification of a tour, so inputs like tour_id and status, extracted from an event template of this event, searches a tour object and updates status attribute of a tour (affecter class). In an event, "Customer cancels a reservation on tour" State Changestype is termination of a reservation, so input like reservation_id (PNR) extracted from an event template of this event, searches a reservation object (affecter class) to be terminated. In an event, "TA Software displays tour details" State Changestype is read a tour, so input like tour_id extracted from an event template of this event, searches a tour object (facilitator object) to be accessed. It places a method display_tour ( ) in a TA software (Initiator) and correspondingly places a method read_tour ( ) in a Tour (Facilitator).In an event, "Employee increase credit limit of customer", State Changestype is calculate, so a new credit limit is calculated and updated for a given customer. An event, "Customer cancels reservation on tour", checks reservation status of a tour booked and triggers an event "TA Software updates booked tour details in a database". In an event, "TA software weekly generates a report of all canceled tours", State Changestype is compute, so all tour-objects that are cancelled are retrieved without modifying their state.

State Changes -to-Association

In an event, "Customer makes reservation on tours", a connection State Changesoccurs between Customers (I) and a Tour (F) so an association is mapped between a Customer (I) and Tour (F) with association name as 'travel'.

State Changes -to-Access

In an event, "Customer views information about tours", a customer (initiator) and a TA Software (facilitator and boundary class) has to have a read_tour( ) method that invokes a selector defined in a tour. A tour is a facilitator and an entity class of an event, so it defines a selector get_tour_details( ) that provides name, source, destination and price of a tour.

State Changes -to-Modifier

In an event, "TA software update availability status of a tour booked", an update method is added to a TA Software (Initiator) and correspondingly add a set status ( ) method to a Tour (Affecter).

Similarly for an event "TA Software updates customer information in a database", write_customer( ) method is added to a TA Software (initiator) that invokes a method set_customer (affecter) in a Customer class.

State Changes -to-Classification

In an event, "Manager provide login ID and password to Employees", a Manager is a type of an Employee so Inheritance relationship can be made between a Manger and an Employee.

Conclusion

The proposed approach helps in identification of events within the system. This exhaustive list is further refined to retrieve external, temporal and system events. Important details about this event can then be stored in event-templates. Event templates can be subject to mapping rules to extract information of classes, their attributes, methods and inter-relationships.

References

[1]Jacobson I. 1992. Object-Oriented Software Engineering, Addison-Wesley.

[2]Jacobson I. 1994. The Object Advantage: Business Process Reengineering with Object Technology, Addison-Wesley.

[3]W. Roy Schulte, "The growing role of Events in Enterprise Applications" Gartner Inc. ,AV-20-3900

[4]K.Singh, Sandeep and Sabharwal, Sangeeta and Gupta, J.P., "Object Oriented Analysis using Event Patterns" in Proceedings of International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering December 3 - 12, 2007,Page 438-442.

[5]Booch, G., "Object-Oriented Analysis and Design, Benjamin", 2nd ed., Cummings, 1994. 155

[6]Dong Liu, Kalaivani Subramaniam, Armin Eberlein, and Behrouz H. Far., "Natural Language Requirements Analysis and Class Model Generation Using UCDA", in Proc. of 17th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2004, Ottawa, Canada, pp.295–304, 2004.

[7]Use Case that works available at www.ocgworld.com/doc/OCG_Use_Cases_that_Work.pdf

[8]Sabharwal, Sangeeta and Gupta, J.P., "An Event-centricmethodology for Generate Class Diagrams ". Journal of Computer Science, Science Publications, USA, ISSN:1549-3636