Journal of Computer Science 6 (11): 1301-1325, 2010
ISSN 1549-3636
© 2010 Science Publications

An Event-Based Methodology to Generate Class Diagrams and its Empirical Evaluation

[1]Sandeep K. Singh, [2]Sangeeta Sabharwal and [1]J.P. Gupta
[1]Department of Computer Science and Engineering and Information Technology,
JIIT A-10 Sector 62, Noida, India
[2]Division of Information Technology, NSIT Sector, 3,
Dwarka, New Delhi, India

Abstract: Problem statement: Event-based systems have importance in many application domains ranging from real time monitoring systems in production, logistics, medical devices and networking to complex event processing in finance and security. The increasing popularity of Event-based systems has opened new challenging issues for them. One such issue is to carry out requirements analysis of event-based systems and build conceptual models. Currently, Object Oriented Analysis (OOA) using Unified Modeling Language (UML) is the most popular requirement analysis approach for which several OOA tools and techniques have been proposed. But none of the techniques and tools to the best of our knowledge, have focused on event-based requirements analysis, rather all are behavior-based approaches. **Approach:** This study described a requirement analysis approach specifically for event based systems. The proposed approach started from events occurring in the system and derives an importable class diagram specification in XML Metadata Interchange (XMI) format for Argo UML tool. Requirements of the problem domain are captured as events in restricted natural language using the proposed Event Templates in order to reduce the ambiguity. **Results:** Rules were designed to extract a domain model specification (analysis-level class diagram) from Event Templates. A prototype tool 'EV-ClassGEN' is also developed to provide automation support to extract events from requirements, document the extracted events in Event Templates and implement rules to derive specification for an analysis-level class diagram. The proposed approach is also validated through a controlled experiment by applying it on many cases from different application domains like real time systems, business applications, gaming. **Conclusion:** Results of the controlled experiment had shown that after studying and applying Event-based approach, student's perception about ease of use and usefulness of OOA technique has significantly improved. Their project reported showed positive feedback about Event-based approach. These results reinforced the evidence that by analyzing events that are likely to happen in a system, one can derive class diagram information from requirements.

Key words: Event meta-model, event template, object-oriented analysis, software requirements engineering, use cases and UML

INTRODUCTION

Event-based systems are rapidly gaining importance in many application domains ranging from real time monitoring systems in production, logistics, medical devices and networking to complex event processing in finance and security. In our day to day life, we often use thermostat, computerized topographical imaging scanner, microwave oven, ECG monitor, cardiac pacemaker and automatic luggage movement system at airport; robots at workplace. All these automated systems have one thing in common that they all fall in the category of event-based systems. We call them as event-based systems due to fact that unlike typical business applications where long descriptive and narrative text is needed to understand the behavior of a system, functionality of these systems can be understood on the basis of events, their flows and interdependencies.

The popularity of event-based system is evident from the fact that an entire book is devoted to complex event processing (Luckham, 2002). The increasing popularity of these event-based systems has opened new challenging issues for them. The issue that this study addresses is to propose a process for requirements analysis of event-based systems and build conceptual

Corresponding Author: Sandeep K. Singh, Department of Computer Science and Engineering and Information Technology, JIIT A-10 Sector 62, Noida, India

models. Conceptual model aids in requirements analysis and understanding of problem domain. Several conceptual modeling tools and techniques have been proposed like Entity-Relationship (ER) model, Extended Entity-Relationship (EER) model, E²R diagram, Higher-Order Entity Relationship Model (HERM), Conceptual Schema Language (CSL), DATAID-1, REMORA methodology, Booch method, Object Modeling Technique (OMT), Object-Oriented Software Engineering (OOSE) and Unified Modeling Language (UML) out of which UML is currently the most popular OO conceptual modeling technique (Luckham, 2002). Rational Unified Process (RUP) is a unified process that proposes rules for effectively using UML for analysis and design (Booch *et al*., 2005; Kruchten, 2003).

After an exhaustive survey of tools and techniques of requirement analysis, we have found to the best of our knowledge that none of the existing approaches or tools have neither focuses on requirements analysis of event based systems nor have used events as basis for requirement analysis and conceptual modeling. Approaches are largely based either on natural language (Abbott, 1983; Jacobson *et al*., 1999; Turk and Vanier, 1993; Coad and Yourdon, 1990; Shlaer and Mellor, 1998; Ross, 1988; Song *et al*., 2005; Ilieva and Ormandjieva, 2005; Mustafa and Awofala, 2004) for which various tools have also been developed (Becker *et al*., 2000; Barber and Graser, 2000; Harmain and Gaizauskas, 2003; Overmyer *et al*., 2001; Wahono and Far, 2002; Drake *et al*., 1993; Perez-Gonzalez *et al*., 2005) or on Use cases (Anda and Sjberg, 2003; Liang, 2003; Liu *et al*., 2003; 2004; Roussev, 2003). NLP based techniques have their own limitations and at the same time Use Cases have been critically reviewed in the recent past (Some, 2005; 2007b; Wiegers, 2005; Ferg, 2003; Samarasinghe and Some, 2005). It is also cited that Use Case modeling is not an effective technique for projects related to data warehouses, batch processing, embedded control software, computationally intensive applications and real time systems (Samarasinghe and Some, 2005).

Due to lack of approaches exclusively for event-based systems and limitations and critical reviews of the existing approaches, we present an iterative approach for requirements analysis of event-based systems by taking events as the starting point. Events have been chosen as a starting point due to several reasons (a) it is more realistic to use events than processes for requirement analysis and conceptual modeling of event-based systems. (b) It is event modeling only that introduces rigor and discipline in Use Case modeling by helping to determine list of Use Cases (Satzinger *et al*., 2006). (c) Events act upon many classes and conversely, the same class may be acted upon by a variety of seemingly unrelated events. Thus, events help the analysts or OO design team to determine which events should be allocated to operations on data centric persistent classes. (d) In carrying out OOA using the parallel conceptual modeling technique i.e., Use Case modeling, a large number of diagrams need to be made before arriving at a final class diagram. Scenarios are extracted from documented Use Case templates to build sequence, activity and collaboration diagrams from which final class diagram is realized. On the contrary, our approach derives analysis level class diagram from events without need to draw other diagrams. (e) Events use a technology-independent stimulus-response modeling technique, while deferring interaction design. (f) Analysis of expected and unexpected events helps to capture the essence of business policy at an early stage of project. Thus, event modeling lets a user create analysis specification that has more value to business in the long run. Due to these arguments in favor of events, this study proposes a novel application of Event modeling in OOA of requirements.

The main contribution of the work is in proposing, validating and automating event-based approach to build analysis level class diagram from the natural language requirements of event-based systems. An Event-Meta model is proposed which forms the foundation for event-based OOA. Requirements are captured in restricted natural language using the proposed Event Templates in order to reduce the ambiguity. Rules are also made to extract a domain model specification (i.e., analysis-level class diagram details) from Event Templates. The proposed approach is validated through a controlled experiment by applying it on many cases from different application domains like real time systems, business applications, gaming. The objective of the controlled experiment is to compare the perceived ease of use and usefulness of the proposed event-based approach with a more conventional and industry standard Use Case based approach. Results have shown that Event-based OOA has improved user's perception significantly. A prototype tool 'EV-ClassGEN' is also developed to provide automation support to extract events from requirements, document the extracted events in Event Templates and implement rules to derive specification for an analysis-level class diagram.

Related work: There are various techniques that have been used in the past to extract components from the requirements for building class and object model.

Techniques proposed have either used natural language processing approach or employed Use cases to identify classes. Some of these approaches have been automated by building their prototype tools.

Techniques and tools based on natural language processing of requirements: Abbott (1983) and Booch *et al*. (2005) proposed a technique that used singular nouns and nouns of direct reference to identify objects and plural and common nouns to identify classes. This approach became the basis for many OOA approaches and tools. Turk and Vanier (1993) have used computerized classification systems and thesauri for the purpose of object oriented analysis of the valid Slovenian earthquake code. In 1991-1992, pioneers like Coad and Yourdon, (1990); Shlaer and Mellor (1988) and Ross (1988) identified certain categories like persons, role and organization, which define application domain entities and help experienced analysts to identify classes or objects. Song *et al*. (2005) have presented a Taxonomic Class Modeling (TCM) methodology that can be used for identification of domain classes during object-oriented analysis in business applications. Ilieva and Ormandjieva (2005), a methodology is proposed for the natural language processing of textual descriptions of the requirements of an unlimited natural language and their automatic mapping to the object-oriented analysis model. Sentences in the text are analyzed and semantic network is built from which OO model (class model) is derived. In another approach (Mustafa and Awofala, 2004), process mapping and clustering techniques from cell manufacturing are used for deriving object-oriented classes from requirements. All the approaches reviewed above points out to basic disadvantages associated with natural language processing of requirements like completeness, accuracy, ambiguity. Therefore, there is always an open need to research on a novel approach to carry out OOA. This has been the motivation for the work carried out in this study.

Several authors have used the techniques described above to develop automation support for the analysts. Some of the popular tools are A Methodology for Automatic Object Identification from System Specification (MOSYS) (Becker *et al*., 2000), Reference Architecture Representation Environment (RARE) (Barber and Graser, 2000), Class Model Builder (CM-Builder) (Harmain and Gaizauskas, 2003), Linguistic assistant for Domain Analysis (LIDA) (Overmyer *et al*., 2001), OOExpert (Wahono and Far, 2002), Automated User Requirements Acquisition (AURA) (Drake *et al*., 1993), A Graphic Object Oriented Analysis Laboratory (GOOAL) (Perez-Gonzalez and Kalita, 2002; Perez-Gonzalez *et al*., 2005).

Techniques based on use cases: In work carried out in (Anda and Sjberg, 2003) authors present a Use case-driven development process for OOA and its validation. However it is reported in empirical findings that this technique leads to problems, such as the developers missing requirements and mistaking requirements for design. A variant of the use case-driven approach is used in which instead of the scenarios the goals of each Use case without descriptions are used to identify classes (Liang, 2003). An approach with a set of artifacts and methodologies, to automate the transition from requirements to detail design is presented in (Liu *et al*., 2003). Roussev (2003), a process is proposed for generating formal object-oriented specifications in OCL and class diagrams from the Use case model of a system through a clearly defined sequence of model transformations.

Liu *et al*. (2004), a methodology and a CASE tool named Use-Case driven Development Assistant (UCDA) is presented to automate natural language requirements analysis and class model generation based on the Rational Unified Process (RUP).

Although Use Case based approaches have been quite popular but several arguments against Use Case have been cited in the literature. Author in (Liu *et al*., 2004) has cited that Use Cases do not alone solve the problem. It is scenario that specifies concrete sequences of actions for the requirement. An overall advantage can be achieved by integration of scenario-based approaches with functional requirements. Even work in (Some, 2005) has emphasized that Use case based requirements engineering approach can be enhanced by integrating Use Case with Scenarios. Another book (Wiegers, 2005) has also cited that Use Case approach is ill suited for projects involving data warehouses, batch processing, hardware products with embedded control software, computationally intensive applications, understanding real time systems, systems that involve complex business rules to make decision and for specifying time triggered function. Work in (Ferg, 2003) has pointed out that Use Case approach discourages the requirements analysts from examining the problem domain, by focusing only on what happens at the system boundary. Several articles have also pointed out problems with Use cases related to understanding, clarity, invisible scope creep; its document centric, time consuming and declarative nature and its inability to differentiate dynamic and static elements of the specification. Although improvisation of Use case based requirements analysis approach have been done (Samarasinghe and Some, 2005; 2006; 2007a; 2007b; Satzinger *et al*., 2006) by either automating Use Case based model generation, improving Use Case Templates or enhancing Use Case based analysis with scenarios but these solutions have not proposed any other alternative, their staring point is still a Use Case.

Due to lack of approaches exclusively for event-based systems and limitations and critical reviews of the existing approaches, we present an iterative approach for requirements analysis of event-based systems by taking events as the starting point. Work described in (Poo, 1999; Muhairat *et al*., 2010) can be considered comparable with our proposed approach. Poo (1999), author has proposed the extension of the Use Case Modeling approach to include business policies modeling.

Events in use cases formed the basis for identifying and specifying classes and business rules. A process known as Event Scripting is used to document event and from it, objects and their relationships are identified. Business rules identified with the events are attached to objects as part of their definitions in class specifications. For each event identified in the Use Case, an Event Script is written. Components of event script like Source, Participants Sets, Pre-Event Conditions and changes help the analysts to identify classes and objects, their attributes, operations and business rules. These identified components form a class specification. Unlike this approach, our work starts directly from events identification, his approach starts from the core step of identifying Use Cases, documenting each Use Case in Use Case Template, then extracting events from the scenarios of Use Case description and documenting each event using the proposed event script. Then from event scripts, class diagram components are extracted. Table 1 gives a critical comparison between Poo's perspective and our proposed approach. Muhairat *et al*. (2010), authors have used traditional event table proposed in McMenamin and Palmer's event partitioning approach (Yourdon, 1988). They have modified the event table to include input message, output message, includes, extends, specializes, destination and source fields and also proposed a five step process to build a class diagram. But our proposed work does not use event table instead we have made an event template based on our event-meta model to document event. Our event template store more detailed information on events in comparison to event table. We have also defined 11 mapping rules to extract class diagram model information from event templates. Unlike their output, our class diagram model information is generated in standardized XMI format to make it importable, so that a class diagram can be made using a UML tool. We have chosen to generate importable XMI file for ArgoUML tool.

Proposed event meta-model: The concept is introduced on events, their types and significance. Significance of an event is explained in terms of what operations are triggered on the participating classes by that event. This conceptual background lays the foundation of Event-based OOA methodology and also helps analysts to identify events from the problem domain. Then proposed Event-Meta Model is presented that has formed the basis for designing event templates.

Event definition and event types: The concept of event itself has been widely used to model software. Our approach is adopted from "Event Partitioning" method that was used to create Data Flow Diagrams (DFDs) during structured analysis of the system (Yourdon, 1988). Their idea has been used for partitioning the requirements during the process of OOA for investigating the role of Events as starting point in OOA of event-based systems.

Technically, an event is a record of system activities with attributes, significance and relativity (Luckham, 2002). "We define event as a happening (occurrence) at a specific time and place that can be described and recorded in the system". Events trigger all the processing in a system, so identifying and analyzing events is a good starting point in requirement analysis.

Table 1: Comparison of perspectives

Poo (1999) approach	Proposed approach
Source of event script in each event is identified from use case description.	Source of event template is directly events identified from requirements, thus eliminating need to first identify Use Cases and write description.
It is an extension to use case modeling.	It is an alternative to use case modeling.
Structural Content of event script is oriented from perspective of defining and understanding business rules and policies.	Structural Content of event templates is oriented from perspective of deriving static and dynamic view that can be modeled in any UML complaint tool.
No temporal or causative relationship is depicted among event scripts.	Event templates can be related by temporal, causative and containment relationship.
Process of Object/Class identification is not formalized.	Our approach is to formalize the process by developing rules to automatically transform event templates to static model.

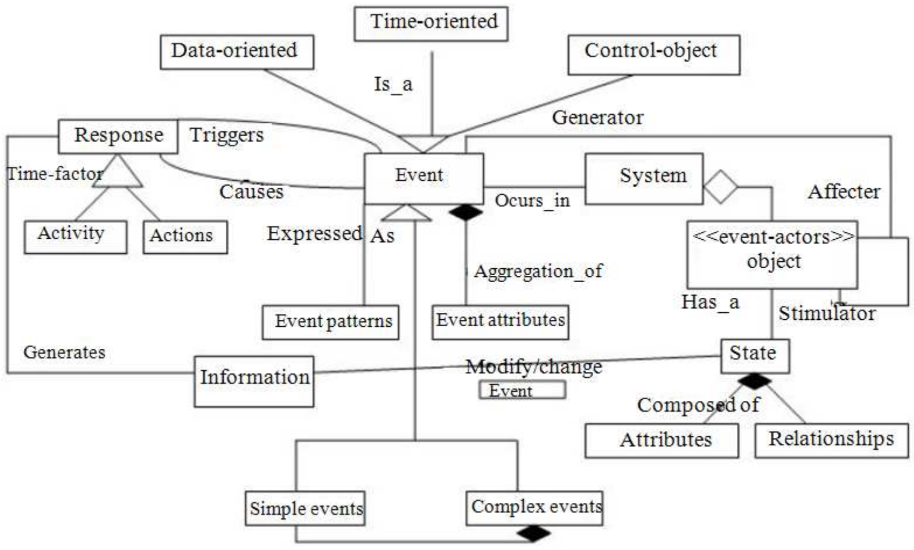


Fig. 1: Event Meta model

Events can affect state of objects (attribute or number of instances), relationships among objects or both. The proposed approach uses three types of events-External events, Temporal events and State events (Satzinger *et al*., 2006; Yourdon, 1988). External event occurs outside the system, usually initiated by the external agent (person or organizational unit or system user). An external agent either supplies or receives data from the system. External events carry data to be exchanged from an external agent to system, from system to an external agent or from one external agent to another in the system. For example, 'Customer places an order'. Here, customer is an external agent and as a result of this event, a new order is generated in the system. Temporal events are generated automatically by the system on reaching a given point of time. They do not occur on fixed date. Time of occurrence could also be relative to some other event occurrence. There is no need for external agent to trigger temporal events. Temporal events include internal or external outputs needed from time to time. For example, 'System produces biweekly payroll'. State events occur when something critical happens inside the system that triggers the need for processing. These events monitor system in order to detect or respond to external system, devices or another object. State events are consequences of external events. For example, Order event in the above example, reduces stock in inventory that results in generating a state event 'Reorder point reached for that product'. Time cannot be predicted for State events. Most of the events in a general application domain are external and temporal. State events are more common in the domain of real time systems. For example in process control system, if vat of chemical is full, then state event, 'turn off the fill valve' is generated.

Event meta-model: An Event-Meta Model has been proposed based on the above concept of events and is shown in Fig. 1. The Event-Meta Model is based on the principle that events are the core elements of event-based systems and causes a system to change its state. Overall functionality of a system is a result of successful execution of chain of events.

Users interact with the system through events. These events trigger the usage in the system (i.e., a Use Case). Using events, analysts can (a) record changes that have occurred over a period of time; (b) identify which object(s) have stimulated events, which object(s) have been affected by events, what operations have made the changes and in which state members. Thus, not only objects but also, changes in their attributes, invocation of specific operations and relationships among objects can also be identified by analyzing the events and participating objects in events. Values changed in events, give idea of attributes of objects.

As shown in Fig. 1, an Event forms the core of our meta-model. Creation and destruction of objects, call to method as well as response from it are all events. There are large numbers of events occurring in a system at a given point of time. Each event is uniquely identified by giving it a unique ID, name and description. These are basic attributes for an Event class in our meta-model. The Meta-Model identifies five types of events-

External oriented, Temporal oriented, State oriented, Simple event and Complex event. Complex events are aggregation of Simple events. Complex event represents higher level abstraction in the event-based system. e.g., Employee administrate customer account is a complex event that includes simple events like employee adds new customer account, employee delete an existing customer account. System is an aggregation of various Objects.

Objects interact and collaborate through events to render the functionality of a system. Every object has a state which is composed up of Attributes and Relationships which an object has with other objects in or outside system. Objects play role of actors in an event. These objects stimulate each other and the stimulus is an event. Object that initiates event plays the role of an Initiator; the one that is affected by event becomes an Affecter and the one that facilitates the occurrence of an event is a Facilitator.

Events have a Response associated with it. Response can be a Use Case that an event triggers or an Action. Response generates information that modifies number of instances, attributes or relationship of an object with other objects. These are called Changes caused by events. Events and Response have a cyclic relationship; an event can trigger a response and a response in turn can generate new events. We call such a set of events as Trigger vector. An event can also be caused by occurrence of some other event (s) and such events are called Causative events. The information required for processing an event is called an Input. Event Meta-Model is the basis of defining the structural contents of an event template used in our proposed event-based methodology.

MATERIALS AND METHODS

Our proposed approach starts from identifying elementary events from the requirements. An elementary event always focuses on an elementary business process. It is performed by one person, at one place, adds a measurable business value and leaves a system in a consistent state. From the pool of events, our proposed methodology generates information for static conceptual model (analysis level class diagram) of the system. The steps are as follows:

- Gather requirements from End user/Domain experts using various modes of information gathering techniques. These requirements can be unstructured text or structured into domain description documents
- Extract elementary events from textual requirements and generate a list of such events. Event list is an exhaustive list of all possible events that have appeared in the textual requirements of the system
- Analyze and identify other new events with the help of either problem domain experts or from events already identified. Categorize each event in different type and discard events that do not fall in any of our described categories
- Formalize and document each event from the list in an Event Template
- Apply mapping rules on event templates to extract information for static model of the system comprising of the candidate classes, their methods, attributes and relationships

The class diagram specification is generated in step 5 is in an importable standardized XMI format. XMI files generated by different UML tools have different tags and are not compatible with each other. E.g. the class diagram XMI file generated by Rational Rose tool is not identical to the one generated by Visual Paradigm. So our approach generates XMI file of class diagram specification for Argo UML tool so that latter on the same XMI file can be used for regenerating the model information in Argo UML tool for the class diagram. A prototype tool ‘EV-ClassGEN’ is also developed to provide automation support to extract events from requirements, document the extracted events in Event Templates and implement rules to derive specification for an analysis-level class diagram. Next we detail the structure of Event Template, its comparison with Use Case template and the proposed rules to derive class diagram specification from event templates.

Event template: In our proposed methodology, events extracted from textual requirements are documented using Event Templates. An Event Template inherits its components from the Event Meta-Model and models every single interaction details of actors with the system. The components of Event-Meta Model are mapped to different fields of an Event Template. The important components of an Event Template are.

Event ID: It is a unique alphanumeric value given to each event identified either directly or indirectly from the requirements. No two events can be assigned same event id. It helps us in tagging detailed description of an event template with its id.

Event description: It is a sentence from the requirements that describes the event identified in Subject-Verb-Object (SVO) pattern. (e.g., customer places an order).

Event name: It is a simple name as extracted from the requirements given in the natural language. This can indicate verb or verb phrases in Subject-Verb-Object (SVO) pattern (e.g., order placed).

Initiator, facilitator or affecter: Initiator starts an event. Facilitator facilitates in the occurrence of an event and Affecter gets affected as a result of execution of an event. Initiator, Facilitator and Affecter are different roles that entities/objects play in different events. There can be single Initiator, Facilitator and Affecter in an event. Different initiators are mapped to different events. Every event has at least one role. An entity can have overlapping roles in events e.g., 'customer' can be initiator in one event and affecter in another event. Ternary relationship involving third entity will have a facilitator otherwise Initiator/ Facilitator roles can even be merged until there is a need to explicitly specify them separately. In example, 'Travel agency stores Tour information', Tour gets created (modified) so Tour is affecter. In example, 'Customer sends complaint through Travel Agency software', Travel agency software is unaffected so it is a facilitator, whereas Customer is an Initiator and Complaint is an Affecter.

Events occur as a chain of related events. An Event triggers other events in a system. Identification of an event in turn helps to identify other events that can be triggered by it. Such related events are called as Causative events and Trigger vector.

Causative events: Causative events of an event are those events that are reasons behind occurrence of that event. While documenting events, focus is on those causative events that are in context of the problem description. Causative events may not be there for events that are triggered independently after system initialization. Time for state events cannot be determined, so causative events play very important role in initiating such events. Trigger Vector: It represents a set of events that are triggered as a result of occurrence of an event. An event can trigger either a single event, set of events that can be executed independently or in parallel. Events relate with other events in event expression using event operators 'event-or', 'event-and', 'event-not' and 'event-xor'. Event-or indicate that either none, one or more than one events can be triggered. Event-xor indicates exactly one event can be triggered. Event-and indicate that all events have to be triggered in parallel.

Event-not indicate non-occurrence (negation) of an event. For example an event e1 ('Customer register with TA software'), triggers an event e2 ('Travel agency provides user id and password to Customer'), e1 is causative event of e2 and e2 is trigger vector of e1. Every event triggered may initiate algorithmically simple or complex services in an object. These services model the behavioral changes in the object. These changes are described in Event Template as Change-event.

Change-event (state changes): An event causes operations to be triggered in participating classes. These operations are side effects of any event on the state of participating entities (Initiator, Facilitator or Affecter). Operations like creation, termination and update (or calculate) change the state of participating objects. While operations like read, access, compute or monitor do not affect state. Hence these operations are described as change-event. E.g., an event 'Customer registers with Travel Agency' causes a change-event 'Creation of Customer Profile object'. These change-event affect classes at different levels like (a) Object level change-event can be- Creation (an object is getting created e.g., Order placed), Termination (an object is getting destroyed e.g., Order cancelled), Read objects (an entire object state gets read from object memory). (b) Attribute level change event can be accessing or updating attributes of objects for performing any calculation, computation and monitoring on that object. Calculations are one that an object performs on its value. Monitoring involves checking of an attribute in an object to detect and respond to external system, device or another object. Computation involves computing a functional value from attributes without modifying an object state. (c) Relationship level change-event can be-A classified B that indicates inheritance relationship (e.g., Order shipped i.e., Order classified as Shipped Order) or a connected B that indicates association relationship (e.g., Person employed i.e., Person is connected/associated to an Organization via is employed relationship).

Timestamp: Events occur at some point of time. Multiple events may occur at the same time and could be unrelated, co-operating, or related with each other. Timestamp records time when a particular event has happened or likely to happen in the system. Since all events in system are related with each other, a relative timestamp value is to be assigned to each event. Assigning the exact timestamps too early at the analysis level is not possible. Thus at the analysis level, dummy timestamp values can be assigned by the analysts while identifying events. The dummy timestamp value can be used in future for reconstructing the sequence of events. Dummy Value assigned to timestamp may be fixed or variable in nature. Variable time stamp indicates that an event occurrence depends on the interaction of user with the system. Variable timestamp is denoted by unique alphanumeric value starting with 'TA' followed by incremental unique id. Higher numerical part of timestamp value indicates a latter occurrence of that event in the system. For example, an event with variable timestamp value TA6 will occur earlier in the system in comparison to an event with timestamp value TA15. Fixed time stamped events indicate periodical events that get initiated after a fixed interval in the system like weekly, monthly, quarterly. Fixed timestamp values are indicated by Daily (FD), Weekly (FW), Fortnightly (FF), Quarterly (FQ) or Annually (FA).

Events with same timestamp value indicate independent events that can occur in parallel in the system. Timestamp for External and Temporal events could be fixed but for State events timestamp is always variable, since the time cannot be determined. Different event templates can be temporally ordered on basis of their timestamp values to map the flow of activities or steps in a scenario. Our approach has not used timestamps so far, instead we have used Trigger vector and Causative events for ordering events.

Inputs/outputs: Whenever a change event occurs in a system, it requires some inputs or generates some outputs. Inputs reflect the data needed for change event whereas output is the data produced from the change event. Input/Output can contribute to describe attributes for an object.

Count: Count in a template indicates the range (minimum to maximum) of number of instances of Initiator, Facilitator and Affecter that can participate in an event. For a given entity count value can be different in different events. Table 2, describes an event template of event "Sensor 1 generate detect signal at start place". Default value of count is 0.1 (zero or 1).

Event template Vs use case template: A comparison of Event Templates is done with Use Case Template (Jacobson *et al*., 1999). The previous sub-section presented different components of an Event Template. Table 3 gives a comparison of Use Case Template (Jacobson *et al*., 1999) with Event Template and highlights essential differences in the two templates.

Proposed rules to derive class diagram specification from event templates: An analysis-level class diagram typically shows attributes and operations; it may show other adornments such as multiplicity and role names as well. The aim of the proposed approach is to derive analysis level class diagram from Event Templates. For this, rules are proposed such that information needed to generate class diagram is extracted from fields of Event Templates. E.g., potential candidate classes name can be extracted from name of Initiator, Facilitator or Affecter; flow of events among Initiator, Facilitator and Affecter helps to determine the message passing sequence; participation of Initiator, Facilitator and Affecter helps to determine the association relationship among them. The type of change-events that happen with an event helps to determine the operations that are to be allocated to class. These rules help the analyst in deriving candidate classes, their stereotypes, attributes, relationships and in placing operations in its appropriate class. These rules are applied on a case study (Jalloul, 2004). Following rules are proposed to derive class diagram specification from Event Templates.

Rule 1 (Class name rule): Every Initiator, Facilitator or Affecter from each event template is mapped to potential candidate classes as each of them is a participating entity in some or the other event. In Event template, a name is specified for every initiator, affecter and facilitator. This name is extracted to make a list of all potential class names. Redundancy in names could be due to synonym or repetition of name. Redundant names of Initiator, Facilitator or Affecter that refer to same entity from real world are merged. Further, refinement in class name can be taken at design level.

Table 2: Event template for event "Sensor 1 generate detect signal at start place

Event ID	EA07	
Event name (verb phrase)	Generate detect signal the package	
Description	Sensor 1 generate detect signal at start place (state/control event)	
Initiator	sensor 1	count
Facilitator	ALCS/Belt 1(start place)	count
Affecter	Signal	count
Timestamp		
Causative events (Preconditions)	EA05	
Inputs	Signal type	
Trigger vector	Sensor 2 generate no-detect signal at scan place	
	Sensor 3 generate no-detect signal at transition place	
	Sensor 4 generate no-detect signal at end place	
Change-event	Connection between Sensor 1 and Signal	

Table 3: Comparison of use case template with our event template

Template component	Use case template	Event template	Comparison
ID	√	√	A unique ID will help in tracing, maintaining and relating event templates of all events in our process.
Name	√	√	Unlike Use Case template that gives a goal-oriented name; event name is interaction oriented. Our process focuses on interactions leading to goals rather than goals in isolation.
Description	√	√	Unlike Use Case description that is a sequence of related events; event description represents a single interaction in system.
Actors (primary and secondary)	√	Initiator facilitator affecter	Unlike Use Case template that differentiate actors in two category-Primary and Secondary Actor;. In event template, three different roles are defined as an Initiator, a Facilitator or an Affecter of event. This gives us three new stereotypes for classes. Facilitator is optional.
Timestamp	√	√	Unlike time information that is specified as a non-functional requirement in Use Case modeling; a timestamp ties an event occurrence with time in the system. It identifies temporal relationships among events; helps to draw an Event Flow diagram and eliminates need for sequence diagram during event-based OOA.
Trigger	√	Causative events	Unlike Use Case where trigger identifies the event that initiates the Use Case; Event Template have causative events which is a set of events that are immediate causes for occurrence of an event.
Pre-conditions	√	Causative events	Unlike pre-conditions in case of Use Case Modeling, Event templates have Causative events that must be executed in system before that event occurs.
Post-conditions	√	Change-event Trigger-vector	Unlike post-conditions in case of Use Case Modeling, Event templates have Trigger vector and Change-events to describe side effects in the system due to execution of an event.
Trigger-vector	×	√	Trigger vector represents event ID's of events that are triggered / caused due to the occurrence of an event.
Inputs/ Outputs	×	√	It represents data that provides input or carries output. Details include entity name and the name of the state members (attributes) and value (content) involved in execution of the event.
Change-event	×	√	Changes in a system are categorized in terms of 13 different Change events as described in Event Template. Every change in our event template records the method name, its types (event category) and which class realizes it.
Normal Flow/ Alternative Flows	√	Event flow diagram	Unlike normal and alternative flow in Use Case template, Event flow is determined through causal or temporal ordering of the t event templates based on timestamp value of each event or using causative and trigger vector of each event template.

Rule 2 (Role name rule): Each entity plays a different role in an event such as a role of Initiator, Facilitator or Affecter. These roles define three new stereotypes that are identified in this approach i.e., Initiator denoted by I, Facilitator denoted by F or Affecter denoted by A. Role of an entity changes with each event. For example, customer plays the role of an initiator in one event and of an affecter in another event. For each entity, all the roles played by it are complied from all event templates and all respective stereotypes are placed in class specification.

Rule 3 (Class type, i.e., Boundary, Entity and Control rule): This rule attaches three class stereotypes with classes identified by Rule 1 and Rule 2. These stereotypes are: Boundary, Control and Entity. Boundary classes are used to model interactions among the system and its entities. Control classes are used to represent coordination, sequencing, transactions and control of other objects. They also encapsulate control related to a specific event. Entity classes are used to model information that is long-lived and often persistent. Initiator and Affecter can be control or entity class. Decision to make a class either one of them is taken at design level. Facilitator rightly acts as a boundary class. For example, in a typical e-marketing system, marketing campaign form, budget system are boundary classes, create marketing campaign is control class and purchased item and customer are entity classes.

Rule 4 (Cardinality rule): Count specified in the Event template is the number of instances of initiator, affecter and facilitator participating in an event. It gives cardinality constraint of the respective entity. Default value of count is 0.1 (zero or 1). Cardinality is not an attribute of a class rather it is a property of association relationship between two or more classes. For example, in an event, customer orders a copy of a catalogue, cardinality of customer in association with catalogue is one. In another event, customer places many orders, cardinality of customer in association with order is many.

Rule 5 (Message passing rule): Messages are passed among Initiator, Facilitator and/or Affecter of events to invoke algorithmically simple or complex methods/ services. Simple services are either to create or destroy an object, read or write object, connect, classify an object with other objects and/or get or set attribute values of an object. Complex services are either calculation that an object performs on its attributes; monitoring that object is responsible for; or value that an object computes from its attributes without modifying them (query). An Initiator of an event initiate messages, facilitator may respond to messages or may facilitate to transfer message to Affecter. An affecter is an end receiver in the message chain. An affecter invokes an appropriate method from its class. Define one function in Initiator and a corresponding response in facilitator or affecter. In response appropriate method of affecter or facilitator is invoked. Rules for specific method invocation are described in Rule 7-10.

Rule 6 (State rule): Events in the system cause change in number of instances (objects), attributes of objects and relationship among objects. Change event field of the Event template describe the type of change that can occur due to an event, corresponding input and output fields describe the attributes getting affected by events for carrying out such a change:

- If the change-event type is creation (e.g., tour created), inputs from input field define new attributes (state members) of an Affecter class
- If the change-event type is termination (e.g., tour terminated), objects already exist, so inputs are attributes of specific affecter (object) to be terminated from the system
- If the change-event type is read or access, objects already exist, so inputs search a specific object to be read or accessed. Such a change-event will not have an affecter instead will have a facilitator
- If the change-event type is modification/updating (e.g., tour modified), objects already exist, so inputs are attributes to be modified/updated or some new attributes to be added to affecter (object)
- Calculate, Compute or Monitor are special cases of modify, read or access
- If the change-event type is calculation, objects already exist, so inputs are attributes to be used to perform some calculations and modify the object state. Output of change event produces important result in the system
- If the change-event type is monitoring, objects already exist, so inputs are attributes to be checked to detect conditions for triggering state or control oriented event. Input is important in this change event to indicate attribute to be monitored. Output of change event produces important result in the system
- If the change-event type is computation, objects already exist, so inputs are attributes used to perform query or compute a functional value without modifying the object state. Output of change event produces important result in the system

Rule 7 (Creation rule): Creation of an object is an vent and so is its destruction. Whenever such events occur, the state of object either gets initialized or destroyed. Such an event occurs with the help of a facilitator of an event. Whenever an event causes a change, such that, change-event type is creation or termination, then an association relationship can be mapped between an Initiator/a Facilitator and an Affecter. A message to create/destroy is passed from an initiator/a facilitator to an affecter. For every object, created or destroyed, a constructor/destructor is added to affecter class and a corresponding create or destroy method is added to an initiator and/or a facilitator that triggers the construction/ destruction of objects. Such methods are given name create/destroy followed by an affecter name. For example event, Customer places an order (creates an order instance) and event, customer cancels an order (destroy an order instance).

Rule 8 (Association rule): 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, 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 change-event type is connection, then an association relationship is mapped between an Initiator and an Affecter or an Initiator and a Facilitator of the event, if not already mapped by earlier rule. This mapping rule affects the class diagram.

In case, the change-event is disconnection, it affects the object diagram. The verb phrase from an event name is mapped to define an association name property of an association. Count attribute specifies cardinality of an association. Association is automatically mapped by rules of creation, access, read, modify and classify rules.

Rule 9 (Access rule): Events that read the state of objects, read through selectors defined in the class. Whenever an event causes a change, such that change-event type is read or access, then an association relationship is mapped between an Initiator and a Facilitator of the event, if not already mapped by earlier rule. This mapping rule does not change the state of the object. The read event is to read the entire state of object whereas access event only reads a part of object's state. This rule adds a selector (overloaded get method) to the 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 the event. Read method is given the name- read followed by name of the facilitator and access method is given the name- access followed by name of attribute of facilitator.

Rule 10 (Modifier rule): Events modify (update) the state of objects through modifiers defined in the class. Whenever an event causes a change, such that change-event type is 'update', then an association relationship is mapped between an Initiator and an Affecter or an Initiator and a Facilitator of the event. The update event only updates a part of an object's state. This rule adds a modifier (i.e., an overloaded set method) to a facilitator (if it is 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 the event. Such methods are given name update followed by name of an affecter.

Rule 11 (Classify rule): If the change-event type 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 the events are mapped to inheritance.

Case study: The proposed Event-based methodology has been applied on several case studies. We describe requirements specification of a single case and its modeling using proposed methodology.

Reservations online case study: A Case study named 'Reservations Online' on object-oriented analysis (Jalloul, 2004) is used to apply our methodology. Following is the description of the user 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 he/she 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".

Modeling using proposed Methodology: Proposed steps are applied on the above case study. After applying steps 1, 2 and 3, following events along with their types are identified from the case. These events are listed in the Table 4a and 4b. Events specified in Table 4a are explicitly specified in the requirements statements whereas events in Table 4b are identified and added by domain expert.

Table 4a: List of events from "reservation online"

List of events automatically identified from requirements:
• Customer view tour information. (External Event)
• Customer makes a reservation on tour. (External Event)
• Customer cancels a reservation on tour. (External Event)
• Customer sends a complaint. (External Event)
• Customer sends a suggestion. (External Event)
• Travel agency keeps tour information through TA software. (External Event)
• TA Software provides user_id and password to customer. (External Event)
• TA Software sends complaint to Travel agency. (Temporal Event)
• TA Software sends suggestion to Travel agency. (Temporal Event)
• Manager provides login_id and password to employee. (External Event)
• Employee adds customer information. (External Event)
• Employees add tour information. (External Event)
• Employees update customer information. (External Event)
• Employees delete customer information. (External Event)
• Employees update tour information. (External Event)
• Employees delete tour information. (External Event)

Table 4b: List of events added by analysts from "reservation online"

List of events added by analysts:

- Customer registers with TA software. (External Event)
- TA software sends a complaint form to the Customer. (Temporal Event)
- TA software sends a suggestion form to the Customer. (Temporal Event)
- TA software generates a monthly report of all potential customers. (Temporal Event)
- TA software monthly sends list of all tours to customers. (Temporal Event)
- TA software weekly generates a report of all booked tours. (Temporal Event)
- TA software weekly generates a report of all canceled tours. (Temporal Event)
- TA software displays the tour details. (Temporal Event)
- 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 in Table 5-7.

Application of rules to case study: The rules were applied to all event templates of events identified in Table 4a and 4b and information to generate class diagram is extracted from event templates. The class diagram specification generated is stored in an importable standardized XMI format for Argo UML tool so that latter on the same XMI file can be used for regenerating the class diagram information. This information is stored in form of XMI specification file which is fed to Argo UML tool to draw class diagram shown in Fig. 2.

Applying rule 1: In the case study chosen, we have identified Initiator, Facilitator and Affecter from all event templates and found the following potential class names (Table 8).

Applying rule 2: In the case study chosen, we have identified the role of Initiator, Facilitator and Affecter from all event templates and found the following Initiators, facilitators and affecters (Table 9).

Applying rule 3: After applying rule 3 we have classified the entities of our case as shown below (Table 10).

Applying rule 4/cardinality rule: In an event, "Customer makes at most four reservations for tour", cardinality of customer (initiator) is not specified so it is assumed to be default '0.1' and for reservation (affecter) the cardinality count is at most four. The cardinality constraint is specified with the class association relationship with other class and is not a property of a class alone.

Applying rule 5/message passing rule: In an event, "Customer makes a reservation on tour", Initiator customer sends a message "create_reservation" to facilitator 'TA software' which sends the same message to affecter 'Reservation'. Affecter 'Reservation', in response, invokes its constructor method in order to create a reservation object.

Applying rule 6(a)/state rule: In an event, "TA Software registers a Customer" change-event type is creation of customer, so inputs like customer name, id, email, phone number and password, extracted from event template of this event, define attribute list of a Customer (affecter class).

Applying rule 6(b)/state rule: In an event, "TA Software update availability status of the tour booked" change event type is modification of tour, so inputs like tour_id and status, extracted from the event template of this event searches a tour object and updates status attribute of a tour (affecter class).

Applying rule 6(c)/state rule: In an event, "Customer cancels a reservation on tour" change-event type is termination of reservation, so input like reservation_id (PNR) extracted from the event template of this event searches a reservation object (affecter class) to be terminated.

Applying rule 6(d)/state rule: In an event, "TA Software displays tour details" change-event type is read tour, so input like tour_id extracted from the event template of this event searches a tour object (facilitator object) to be accessed. It places method display_tour ( ) in TA software (Initiator) and correspondingly places method read_tour (in Tour (Facilitator).

Applying rule 6(e)/state rule: In an event, "Employee increase credit limit of customer", change-event type is calculate, so new credit limited is calculated based on some criteria and updated for a given customer instance.

Applying rule 6(f)/state rule: An event, "Customer cancels reservation on tour", checks reservation status of tour booked and triggers an event "TA Software updates booked tour details in database".

Applying rule 6(g)/state rule: In an event, "TA software weekly generates a report of all canceled tours." change-event type is compute, so all tour-objects that are cancelled are retrieved without modifying their state.

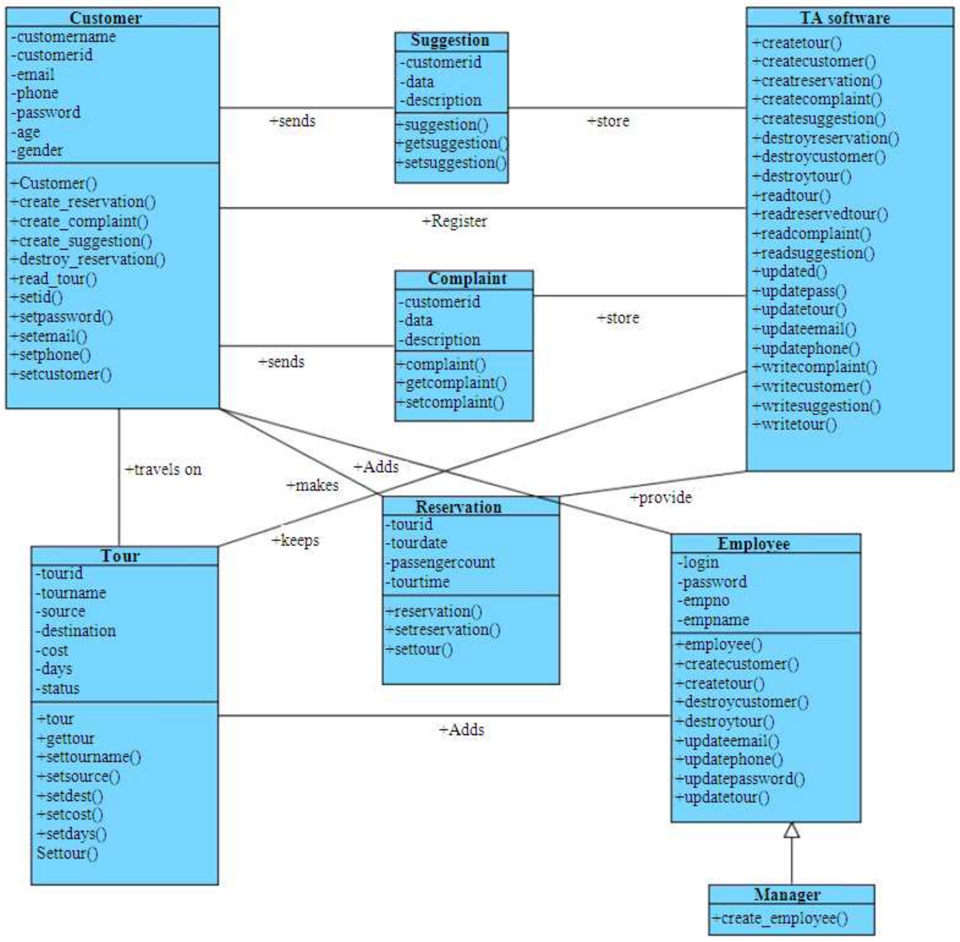


Fig. 2: Class diagram using event-based approach

Table 5: Event template for event "travel agency keeps tour information through TA software"

Event ID	EA03	
Event	Name keeps tour information	
Description	Travel agency keeps tour information through TA software	
Initiator	NULL	Count
Facilitator	Travel agency Software	Count
Affecter	Tour	Count
Timestamp	TA2	
Causative events (preconditions)	EA01	
Inputs	Tour id, Tour name, Source, destination, cost, days, availability status.	
Trigger	Vector NULL	
Change-event	Connection event between Travel agency and tour	
Creation event of	Tour (Tour class)	

Table 6: Event template for event "customer registers with TA software"

Event ID	EA04	
Event	Name register customer	
Description	Customer registers with TA software	
Initiator	NULL	Count
Facilitator	TA Software	Count
Affecter	Customer	Count
Timestamp	TA3	
Causative events (preconditions)	EA02	
Trigger	Vector Travel agency provide user_id and password to customer	
Inputs	Customer name, ID, email, phone number, password	
Change-event	Creation event of Customer profile	

Table 7: Event template for event "travel agency provide user_id and password to customer"

Event ID	EA05	
Event	Name Provides login_id and password	
Description	Travel agency provide user_id and password to customer	
Initiator	Travel agency Software	Count
Facilitator	Null	Count
Affecter	Customer	Count
Timestamp	TA4	
Causative events (preconditions)	EA04	
Inputs	User_id and Password	
Trigger	Vector Customer view tour information	
	Customer make a reservation on tour	
	Customer send a complaint	
	Customer send a suggestion	
Change-event	Update Customer profile	

Table 8: List of potential class names

Customer
Tour
Travel agency*
Reservation
Travel agency software*
Complaint
Suggestion
Manager
Employee

Table 9: List of potential classes with our stereotypes

Customer	Initiator, affecter
Tour	Facilitator, affecter
Reservation	Facilitator, affecter
Travel	Agency software initiator, facilitator
Complaint	Facilitator or affecter
Suggestion	Facilitator or affecter
Manager	Initiator
Employee	Initiator, affecter

Table 10: List of potential classes with UML stereotypes

Customer	Entity class
Tour	Entity class
Reservation	Entity class
Travel agency software	Boundary, control class
Complaint	Entity class
Suggestion	Entity class
Manager	Boundary class
Employee	Entity class

Applying rule 7/creation rule: Customer object is getting created in system with event "TA Software registers customer", constructor for customer is added to affecter(Customer) and corresponding create_customer( ) method is added to a facilitator(TA Software). Similarly for event "Customer cancels reservation", reservation object is destroyed, so a destructor is added to an affecter (reservation class) and a corresponding method destroy_reservation( ) is added to an initiator (Customer) and a facilitator(TA Software).

Applying rule 8/Association rule: In an event, "Customer makes reservation on tours", a connection change event occurs between Customer (I) and Tour (F) so an association is mapped between a Customer (I) and Tour (F) with association name as 'travel'.

Applying rule 9/access rule: In an event, "Customer view information about tours", a customer class (initiator) and TA Software (facilitator and boundary class) has to have a read_tour( ) method that invokes a selector defined in the 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.

Applying rule 10/modifier rule: In an event, "TA software update availability status of the tour booked", an update method is added to TA Software (Initiator) and correspondingly add set status ( ) method to Tour (Affecter). Similarly for event "TA Software updates customer information in database", write_customer( ) method is added to TA Software (initiator) that invokes set_customer (affecter) in Customer class.

Applying rule 11/classify rule: In an event, "Manager provide login ID and password to Employees", Manager is a type of Employee so Inheritance relationship can be made between Manger and Employee.

Figure 2 shows the class diagram generated as a result of applying rules to all the documented event templates of Reservations Online Case Study.

A prototype tool 'EV-ClassGEN' (Fig. 3) is developed in Java to provide automation support to (a) extract events from requirements, (b) document the extracted events in Event Templates and (c) implement rules to derive specification for an analysis-level class diagram. The tool has a modular structure that takes textual requirements specification as input. The three modules are (a) E-XTRACTOR (b) Event Template Generator and (c) Class Diagram Generator.

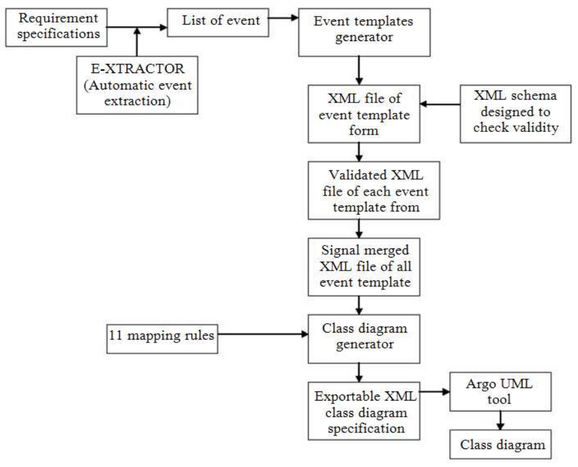

Requirement specifications
List of event
Event templates generator
E-XTRACTOR (Automatic event extraction)
XML file of event template form
XML schema designed to check validity
Validated XML file of each event template from
Signal merged XML file of all event template
11 mapping rules
Class diagram generator
Exportable XML class diagram specification
Argo UML tool
Class diagram

Fig. 3: EV-ClassGEN Tool Architecture

E-XTRACTOR is a domain independent module that automates the process of identification, extraction, analysis and categorization of events. E-XTRACTOR module uses a systematic approach based on Subject, Verb and Object (SVO) pattern is used to extract and formalize events from textual requirements expressed in English as a natural language. Subject-verb-object pattern identifies an event in a sentence. E.g. consider the sentences which are events in typical order processing system, "Customer places order", "Sales Manager denies credit request", "Marketing Department changes prices". Nearly every complete sentence has at least a verb and subject. Some of the commonly used sentence patterns that are used to identify events are SUBJECT-VERB (Coyotes howl), SUBJECT-VERB-OBJECT (Elephants frighten mice), SUBJECT-VERB-INDIRECT OBJECT-DIRECT OBJECT (Mary baked Fred a cake). There is a presence of Subject, Verb and Object in all of them. It also gives the users benefit to analyze, classify and refine the list of automatically extracted events. User can further add new events that are not explicit in the requirements. In order to evaluate the performance of the module, two coverage metrics are also proposed-Coverage metric and Coverage Accuracy metric. These metrics compare events generated by the module with events extracted manually by domain experts from the case studies. Coverage metric is defined as the percentage of total number of events extracted by our E-XTRACTOR over total number of events manually extracted by domain experts. Coverage Accuracy metric is defined as the percentage of correct events extracted by our E-XTRACTOR over total number of events extracted by our E-XTRACTOR. The module has been tested on several case studies from different domains and has shown very promising results (Singh *et al*., 2009a). The module takes input as natural language textual requirements written in English and gives the output in textual format (Fig. 4). Word outside parenthesis represents event while the arguments of events are represented inside parenthesis. Arguments represent subject, object and context information of an event. It uses Stanford's Part of Speech (POS) tagger to generate tagged output. We have implemented 15 parsing rules that are applied on the output of the POS tagger to automatically extract list of SVO patterns (Events) from the textual requirements in XML format (Fig. 5). XML format shows list of events embedded in Event_list root tag. Inside root tag a triplet of <Subject, Verb, Object> is specified as child elements. Parsing rules used to extract events are described in detail in (Singh *et al*., 2009b). Once list of Events and their types is finalized, then it is passed on to Event Template Generator module.

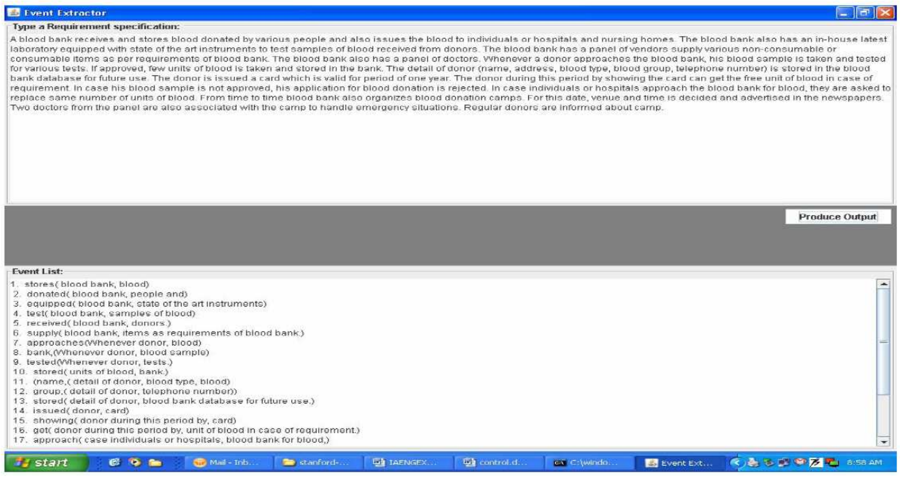
Event Template Generator module document each event from the final list of events using Event Template (Fig. 6) and store the output as validated XML file. For validating the event template XML file, an XML schema is designed that reads the contents of XML file and generate a validated XML File. Tool merges all the XML files corresponding to different event templates in one single file which is used by Class Diagram Generator module.

Class Diagram Generator module implements the proposed 11 mapping rules to generate an importable XML class diagram specification, in XML Metadata Interchange (XMI) format for Argo UML tool. On importing the XMI file containing class diagram model information, Argo UML tool shows the entire model information (class names, associations, generalization, operations and attributes) as tree like structure (Fig. 7).

Controlled experiment:

Objective of the experiment: A lot of empirical work has already been done for validating an approach or a hypothesis. The experimental setup presented here is inspired from various approaches as described in (Cheong, 2008; Dritsakis, 2004; Dritsaki and Adamopoulos, 2005; Dritsakis and Gialetaki, 2005; Fang and Liu, 2007; Sharahili and Liu, 2008; Xu *et al*., 2007).

There are many approaches to generate class diagram specification from the requirements that can be used for comparisons with the proposed approach, but we have used industry standard Use Case approach. The objective of conducting controlled experiment is to compare effectiveness of the conventional Use Case based approach that already exists with Event-based approach that we have proposed for generating class diagram specification from the requirements.



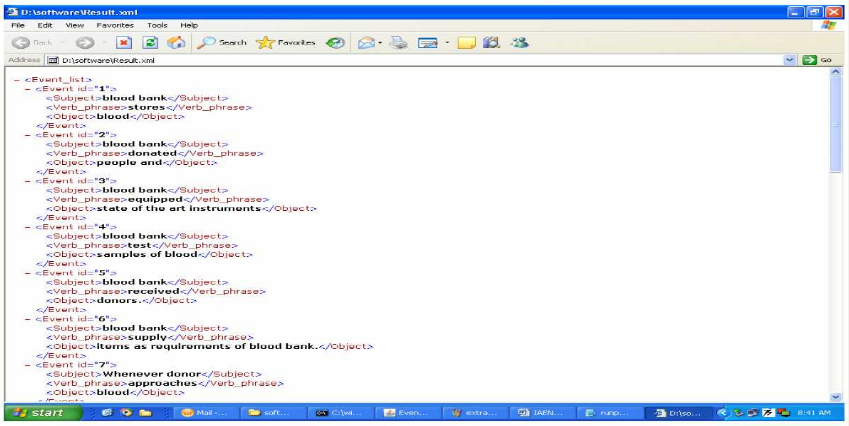
Type a Requirement specification:

A blood bank receives and stores blood donated by various people and also issues the blood to individuals or hospitals and nursing homes. The blood bank also has an in-house latest laboratory equipped with state of the art instruments to test samples of blood received from donors. The blood bank has a panel of vendors supply various non-consumable or consumable items as per requirements of blood bank. The blood bank also has a panel of doctors. Whenever a donor approaches the blood bank, his blood sample is taken and tested for various tests. If approved, few units of blood is taken and stored in the bank. The detail of donor (name, address, blood type, blood group, telephone number) is stored in the blood bank database for future use. The donor is issued a card which is valid for period of one year. The donor during this period by showing the card can get the free unit of blood in case of requirement. In case his blood sample is not approved, his application for blood donation is rejected. In case individuals or hospitals approach the blood bank for blood, they are asked to replace same number of units of blood. From time to time blood bank also organizes blood donation camps. For this date, venue and time is decided and advertised in the newspapers. Two doctors from the panel are also associated with the camp to handle emergency situations. Regular donors are informed about camp.

Event List:

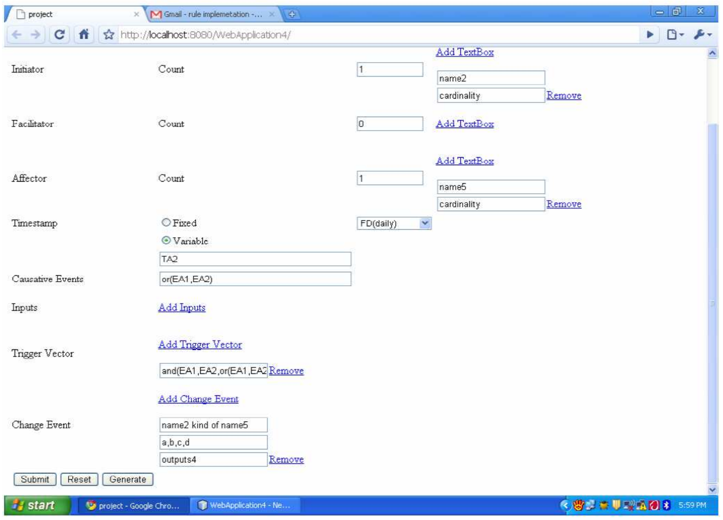
```
1. stores( blood bank, blood)
2. donated( blood bank, people and)
3. equipped( blood bank, state of the art instruments)
4. test( blood bank, samples of blood)
5. received( blood bank, donors.)
6. supply( blood bank, items as requirements of blood bank.)
7. approaches(Whenever donor, blood)
8. bank,(Whenever donor, blood sample)
9. tested(Whenever donor, tests.)
10. stored( units of blood, bank.)
11. (name,( detail of donor, blood type, blood)
12. group,( detail of donor, telephone number))
13. stored( detail of donor, blood bank database for future use.)
14. issued( donor, card)
15. showing( donor during this period by, card)
16. get( donor during this period by, unit of blood in case of requirement.)
17. approach( case individuals or hospitals, blood bank for blood,)
```

Fig. 4: E-XTRACT tool


D:\software\Result.xml
File Edit View Favorites Tools Help
Back Search Favorites
Address D:\software\Result.xml
Go

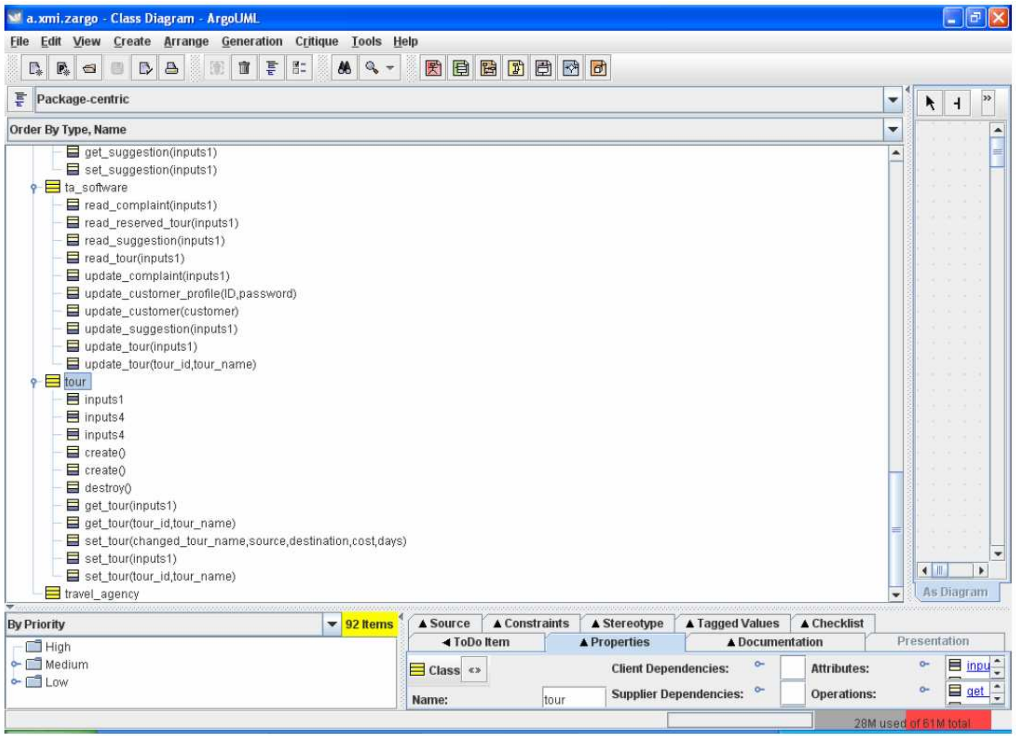
```
- <Event_list>
  - <Event id="1">
      <Subject>blood bank</Subject>
      <Verb_phrase>stores</Verb_phrase>
      <Object>blood</Object>
    </Event>
  - <Event id="2">
      <Subject>blood bank</Subject>
      <Verb_phrase>donated</Verb_phrase>
      <Object>people and</Object>
    </Event>
  - <Event id="3">
      <Subject>blood bank</Subject>
      <Verb_phrase>equipped</Verb_phrase>
      <Object>state of the art instruments</Object>
    </Event>
  - <Event id="4">
      <Subject>blood bank</Subject>
      <Verb_phrase>test</Verb_phrase>
      <Object>samples of blood</Object>
    </Event>
  - <Event id="5">
      <Subject>blood bank</Subject>
      <Verb_phrase>received</Verb_phrase>
      <Object>donors.</Object>
    </Event>
  - <Event id="6">
      <Subject>blood bank</Subject>
      <Verb_phrase>supply</Verb_phrase>
      <Object>items as requirements of blood bank.</Object>
    </Event>
  - <Event id="7">
      <Subject>Whenever donor</Subject>
      <Verb_phrase>approaches</Verb_phrase>
      <Object>blood</Object>
```

Fig. 5: Output in XML format


project
Gmail - rule implemetation - ...
http://localhost:8080/WebApplication4/
Submit
Reset
Generate

Initiator	Count	1	Add TextBox name2 cardinality Remove
Facilitator	Count	0	Add TextBox
Affector	Count	1	Add TextBox name5 cardinality Remove
Timestamp	○ Fixed ● Variable TA2	FD(daily)	
Causative Events	or(EA1,EA2)		
Inputs	Add Inputs		
Trigger Vector	Add Trigger Vector and(EA1,EA2,or(EA1,EA2 Remove		
Change Event	Add Change Event name2 kind of name5 a,b,c,d outputs4 Remove		

Fig. 6: EV-ClassGEN tool event template GUI interface



```
        get_suggestion(inputs1)
        set_suggestion(inputs1)
    ta_software
        read_complaint(inputs1)
        read_reserved_tour(inputs1)
        read_suggestion(inputs1)
        read_tour(inputs1)
        update_complaint(inputs1)
        update_customer_profile(ID,password)
        update_customer(customer)
        update_suggestion(inputs1)
        update_tour(inputs1)
        update_tour(tour_id,tour_name)
    tour
        inputs1
        inputs4
        inputs4
        create()
        create()
        destroy()
        get_tour(inputs1)
        get_tour(tour_id,tour_name)
        set_tour(changed_tour_name,source,destination,cost,days)
        set_tour(inputs1)
        set_tour(tour_id,tour_name)
    travel_agency
```

```
High
Medium
Low
```

```
Class   Client Dependencies:        Attributes:   inpu
Name: tour   Supplier Dependencies:        Operations:   get
28M used of 61M total
```

Fig. 7: Class diagram specification from EV-ClassGEN tool in argo UML

Effectiveness is measured in terms of perceived ease of use and perceived usefulness of an approach by the user (Davis, 1989). Through this experiment, we wanted to empirically conclude that there is a difference in perceived ease of use and perceived usefulness in using Event-based approach vis-a-vis conventional approach, from the viewpoint of users.

Experimental design and setup: We have conducted a controlled experiment using two-group posttest-only randomized experiment. The posttest only randomized experimental design has simple structure and is one of the best research designs for assessing cause-effect relationships. As in our case, we are measuring effect in terms of improvements in perceived ease of use and perceived usefulness of the two groups, after training them on two different conceptual modeling approaches, so we have used this experimental design. It is easy to execute and, because it uses only a posttest, it is relatively inexpensive.

From around 480 undergraduate students, 160 voluntaries were chosen randomly from B. Tech II year and B. Tech Final year, from two different courses, Object-Oriented Programming and Software Quality respectively. The final year students were chosen to represent professional practitioners. All the selected subjects have volunteered to participate in this activity. The groups were randomly assigned tasks. Experimental group got the training on Event-based approach while the Control group (the comparison group) got the training on Use Case based approach to avoid threat to validity. None of the groups knew about the hypothesis. Documented project reports were collected from all the users. Measurement of perceived ease of use and perceived usefulness was collected on the basis of 12 parameters. These parameters were rated on Likert’s seven point scale. The measurements taken, as well as an assessment of the completed project reports, were evaluated and statistically analyzed to investigate difference in perceived ease of use and perceived usefulness in using Event-based vis-a-vis Use Case based approach.

The experiment was conducted in two phases. Subjects were divided into Experimental and Control groups, each consisting of 80 students. Special care was taken to make sure that subjects get assigned to any one group only without overlapping. First phase was on Concept Teaching where both Event-based and Use Case based approach was taught to experimental group and control group respectively. After a week, second Phase Concept Application was conducted with all subjects, where 30 case studies were randomly distributed to all users of both experimental as well as control groups. Every student worked independently, so with 30 systems, we had 80 unique data points in each group which is adequate to perform statistical tests. After applying respective approaches, they were asked to extract information to generate class diagrams from specification. Below are details of the components of the controlled experiment:

- Independent variable: The independent variables are the two Object-Oriented Analysis approaches (Conventional vis-à-vis Event-based) used for deriving class diagram specification from the requirements
- Dependent variable: There are two dependant variables in our controlled experiment whose effect is to be measured in the context of independent variables. These are-Perceived usefulness and Perceived ease of use. Perceived Usefulness is defined as ‘‘the degree to which a person believes that using a particular approach would enhance his or her job performance’’ whereas Perceived ease of use refers to ‘‘the degree to which a person believes that using a particular approach would be free of effort
- Context variables: The effect of a specific technique will depend on the context in which it is used. The important context variables in our controlled experiment are subjects and task:
    - Subjects: In our case subjects were 160 Undergraduate students from B. Tech II year and B. Tech Final year. These groups of subjects consequently represent our target population
    - Task: The task of the experiment was to first extract information for class diagram and then construct a class diagram for case study. The subjects received a textual requirements document along with detailed rules on how to apply conventional and Event-based approach. Controlled group used Noun and Use case based OOA approaches. Control Group was given rules for Use Case Modeling and in writing effective Use Cases from (Jacobson *et al.*, 1999) whereas the experimental groups were given rules based on the proposed approach. The amount of information is kept same in both the rules, to avoid threats to validity
- Material: During controlled experiment, subjects were given the following material
    - Requirements document S
    - Use Case Template Sample (Filled and Blank format)
    - Event Template Sample (Filled and Blank format)
    - Stationary items like Pen, Pencil and Blank sheets
    - Questionnaires: After completing Object-Oriented Analysis of case study, using the approach assigned to them, we asked student’s opinion on perceived usefulness and perceived ease of use of OOA approach using a questionnaire that has 12 parameters. The subjects were asked to mark the score against each parameter, according to a Likert-type seven-point response format where 1 indicates ‘‘strongly agree,’’2 indicates ‘‘moderately agree,’’ 3 indicates ‘‘slightly agree,’’ 4 indicates ‘‘neutral,’’5 indicates ‘‘slightly disagree,’’ 6 indicates ‘‘moderately disagree,’’ and 7 indicates ‘‘strongly disagree.’’ The perceived usefulness and perceived ease of use questionnaires consist of six parameters each with their respective acronym given inside parenthesis. For perceived usefulness, the questions are (PU1) accomplishes requirement analysis more quickly, (PU2) improves requirement analysis performance, (PU3) increases productivity in requirement analysis, (PU4) enhances effectiveness in requirement analysis, (PU5) makes it easier to do requirement analysis and (PU6) useful in requirement analysis. For perceived ease of use, the six parameters are (PE1) need to consult modeling manual and/or reference, (PE2) easy to model what I want to, (PE3) easy to understand, (PE4) rigid and inflexible to understand, (PE5) easy to remember how to do requirement analysis and (PE6) easy to use. This questionnaire gave us the student’s perception of using Event-based approach vis-à-vis conventional approach

Case study/systems: We took around 30 case studies from different application domains. They range from simple to complex cases. All case studies were such that any of the two approaches can be easily applied without alerting the descriptions. Some of the titles are Implementation of wave optics, UEFA champions League, Resort Management System, Fighter Plane control System, University database management system, LIC management System, Desktop window management, Mall Management system, ATM System, Metro management system, KIT management system, Monopoly board game, Connect-stay connected, Airport Management system, Online auction system, Football penalty shoot, Image editor, E-stock.com, Tutorial on DS, Business Game, Graph Plotter, Chess, Ludo, KIT Counselling, Snake, Brainvita, Solitaire to name a few.

Data collection: In the end of activity, we collected case studies, filled templates forms (Use Case template as well as Event template), list of Events, Use Cases and Class diagrams drawn using the approach assigned to them in the respective session and project reports from all users.

Research hypothesis and test of hypothesis: We expected that the users will find Event-based approach more effective in terms of perceived ease of use and usefulness as compared to the conventional approach. Our null hypothesis is that: There is no significant difference between the student's perception of ease of use and usefulness when following an Event-based approach and when following a conventional approach. To compare Perceived Ease of Use and Usefulness of Event-based approach and Conventional approach following hypotheses were tested in controlled experiment using two-tailed paired t-test. The t-test was used as we have to assess whether the means of two groups are statistically different from each other. If they are different then, is the difference positive or negative. So our results of statistical tests can go in either direction. Therefore, we have used two-tailed t-test. Moreover, two tailed t test is also appropriate for the analysis of the posttest-only two-group randomized experimental design that we have chosen for our experiment. The important points when one considers doing a t-test on a Likert scale question is that a Likert scale question with only 5 possible answers may not possibly possess a normal probability distribution. This is because the range of answers is discrete, not continuous (presumably one is not allowed to answer 1.3 or 2.55). In order to check the distribution, we have plotted frequency results of our questions using a scatter diagram and found the distribution is mound shaped. Therefore, it was approximated as a normal distribution:

$H_0$1: There is no difference in the perception of subjects with respect to PU1 about two approaches

$H_0$2: There is no difference in the perception of subjects with respect to PU2 about two approaches

$H_0$3: There is no difference in the perception of subjects with respect to PU3 about two approaches

$H_0$4: There is no difference in the perception of subjects with respect to PU4 about two approaches

$H_0$5: There is no difference in the perception of subjects with respect to PU5 about two approaches

H_{06}: There is no difference in the perception of subjects with respect to PU6 about two approaches

$H_0$7: There is no difference in the perception of subjects with respect to PE1 about two approaches

$H_0$8: There is no difference in the perception of subjects with respect to PE2 about two approaches

$H_0$9: There is no difference in the perception of subjects with respect to PE3 about two approaches

$H_0$10: There is no difference in the perception of subjects with respect to PE4 about two approaches

$H_0$11: There is no difference in the perception of subjects with respect to PE5 about two approaches

$H_0$12: There is no difference in the perception of subjects with respect to PE6 about two approaches

RESULTS AND DISCUSSION

Results of descriptive and inferential techniques: Analysis of data collected was done by a group of faculty members. None of the authors were involved in evaluating the results in order to avoid threat to validity. Table 11-14 show detailed descriptive statistics of both the approaches. Figure 8 shows comparative mean of the two approaches on the basis of 12 parameters.

Table 11: Descriptive statistics of Perceived Usefulness (PU)

	N	Minimum	Maximum	Mean	SD
Conventional PU1	80	1	7	2.938	1.3626
Event-based PU1	80	1	5	1.900	1.0140
Conventional PU2	80	1	7	2.590	1.3570
Event-based PU2	80	1	5	1.730	0.8260
Conventional PU3	80	1	6	2.540	1.2010
Event-based PU3	80	1	5	2.340	0 7110
Conventional PU4	80	1	6	2.750	1.2880
Event-based PU4	80	1	6	1.850	1.2440
Conventional PU5	80	1	6	2.710	1.3800
Event-based PU5	80	1	5	1.550	1.0660
Conventional PU6	80	1	6	2.660	1.2720
Event-based PU6	80	1	5	1.560	0.9390

Table 12: Descriptive statistics of Perceived Ease of use (PE)

	N	Minimum	Maximum	Mean	SD
Conventional PE1	80	1	7	3.19	1.654
Event-based PE1	80	1	7	3.91	2.076
Conventional PE2	80	1	6	3.05	1.377
Event-based PE2	80	1	6	2.00	1.125
Conventional PE3	80	1	6	2.89	1.414
Event-based PE3	80	1	5	1.80	1.152
Conventional PE4	80	1	7	4.09	1.663
Event-based PE4	80	2	7	5.75	1.196
Conventional PE5	80	1	6	2.83	1.251
Event-based PE5	80	1	6	2.93	1.199
Conventional PE6	80	1	7	3.04	1.453
Event-based PE6	80	1	5	2.20	1.130

Findings of controlled experiment and project reports of subjects: Perceived Usefulness of approach: From the descriptive statistics of Table 11, we can clearly see that mean difference between conventional and Event-based approach is positive for all 6 parameters of Perceived Usefulness which means that subjects are more towards agreement that Event-based approach has better perceived usefulness than Conventional approach. Paired t-test results in Table 13 indicate that there is a significant difference in terms of 5 out of 6 parameters between Event-based and conventional approach, in carrying out OOA from requirements. Only with respect to parameter PU3 difference between two approaches is found insignificant. This is also validated by the minimum value of mean difference calculated for PU3. This indicates that subjects believe that there is no significant difference between event-based and Use Case based approaches with respect to their ability to increase productivity in requirements analysis. At the same time they believe that event-based approach is better than conventional approach in accomplishing requirement analysis more quickly, in improving requirement analysis performance, in enhancing effectiveness in requirement analysis and in making easier to do requirement analysis and is more useful in requirement analysis. Thus we reject hypothesis H_01, H_02, H_04, H_05, H_06 and accept H_03. So there is no difference w.r.t. PU3 (increases productivity in requirement analysis) in two approaches.

Perceived ease of use of approach: From the descriptive statistics in Table 12, we can clearly see that mean difference between conventional and Event-based approach is positive for 3 parameters of Perceived Ease of Use (PE2, PE3 and PE6). The mean difference between conventional and Event-based approach is negative for 3 parameters of Perceived Ease of Use (PE1, PE4 and PE5). Opinion regarding PE4 and PE1 is actually favorable for Event-based approach. Paired t-test results in Table 14, indicate that there is a significant difference in terms of 5 out of 6 parameters between Event-based and conventional approach, in carrying out OOA from requirements. Only with respect to parameter PE5 difference between two approaches is found insignificant. This indicates that subjects believe that there is no significant difference between event-based and Use Case based approaches with respect to their ability to remember how to do requirement analysis. At the same time it indicates that subjects believe that event-based approach is easy to model, understand and use. Event-based approach is not rigid and inflexible to understand and they do not need to consult modeling manual and/or reference. This is also validated by the minimum value of mean difference calculated for PE5. Thus we reject hypotheses H_07, H_08, H_09, H_010, H_012 and accept H_011. So there is no difference w.r.t PE5 (easy to remember how to do requirement analysis).

In the controlled group, for carrying out a detailed analysis of Use cases, activity, sequence and collaboration diagrams were made following which an initial class model was derived from the requirements.

Then, sequence and collaboration diagrams were made to reveal the dynamic behavior of the system in terms of dynamic interactions among and within objects. Sequence and collaboration diagrams are useful as a basis for object design as well as method design. Many useful methods could be identified and derived from incoming and outgoing messages in these diagrams. In contrast, in Event-based approach, a detailed class diagram was derived from events as starting point and they did not focus on any other diagram. The 11 rules described in the approach helped them to determine which events should be allocated to operations on data centric persistent classes. Attributes, methods and associations with cardinality were easily identified. Thus, we can say that taking events as starting point in OOA, helps to derive analysis-level class diagram from requirements. The findings of the controlled experiment reinforced the evidence that Event-based approach has brought a significant change in perception of users about using OOA technique.

Threats to validity: Validity is the best available approximation to the truth of a given proposition, inference, or conclusion. We discuss threats to the conclusion, construct, internal and external validity with respect to our controlled experiment. Our goal is firstly to help the readers qualify the results that are presented in this study and secondly, propose future research by highlighting some of the issues associated with our study.

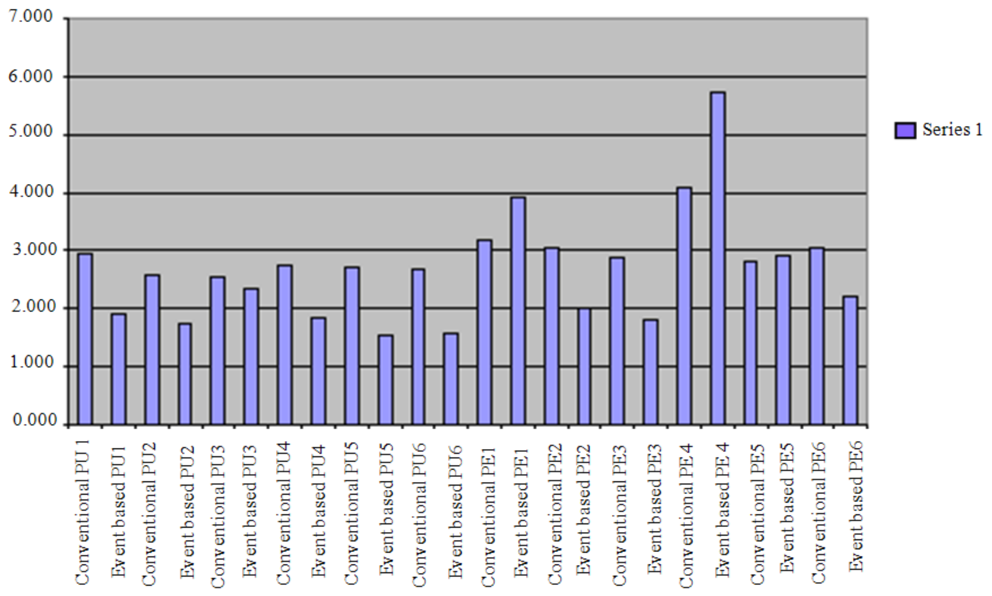


Fig. 8: Comparative mean of conventional and event-based approaches on 12 parameters

Table 13: Paired t-test results of perceived usefulness

	Paired difference						
Lower upper	Std. Mean	Std. deviation	error mean	95% confidence interval of the difference	t	df	Sig. (2-tailed)
Conventional PU1-Event-based PU1	1.0375	1.6416	0.1835	0.6722	1.4028	5.653	79.000
Conventional PU2-Event-based PU2	0.8630	1.5970	0.1790	0.5070	1.2180	4.831	79.000
Conventional PU3-Event-based PU3	0.2000	1.4790	0.1650	-0.1290	0.5290	1.210	79.230
Conventional PU4-Event-based PU4	0.9000	1.7690	0.1980	0.5060	1.2940	4.551	79.000
Conventional PU5-Event-based PU5	1.1630	1.7460	0.1950	0.7740	1.5510	5.954	79.000
Conventional PU6-Event-based PU6	1.1000	1.6660	0.1860	0.7290	1.4710	5.907	79.000

Table 14: Paired t-test results of perceived ease of use

	Paired differences						
Lower upper	Mean	Std. deviation	Std. error mean	95% Confidence interval of the difference	t	df	Sig. (2-tailed)
Conventional PE1-Event-based PE1	-0.725	2.392	0.267	-1.257	-0.193	-2.7110	79.008
Conventional PE2-Event-based PE2	1.050	1.848	0.207	0.639	1.461	5.0820	79.000
Conventional PE3-Event-based PE3	1.088	1.752	0.196	0.698	1.477	5.5530	79.000
Conventional PE4-Event-based PE4	-1.663	2.092	0.234	-2.128	-1.197	-7.1060	79.000
Conventional PE5-Event-based PE5	-0.100	1.747	0.195	-0.489	0.289	-.5120	79.610
Conventional PE6-Event-based PE6	0.838	1.965	0.220	0.400	1.275	3.8130	79.000

Conclusion validity: Conclusion validity is the degree to which conclusions can be drawn about the existence of a statistical relationship between treatments and outcomes. In our controlled experiment, we have treated the experimental group by teaching them Event-based approach and tried to measure their change in perceptions as outcomes. We have avoided low reliability threat by taking from around 480 undergraduate students 160 voluntaries as subjects, randomly. These were from B. Tech II year and B. Tech Final year from two different courses, Object-Oriented Programming and Software Quality respectively. The groups were randomly assigned tasks. To avoid threat due to poor reliability of treatment implementation, both conventional and Event-based approaches were taught in similar manner by same faculty in special lecture sessions. Both approaches were allotted equal number of contact hours. To avoid threat due to random irrelevancies in the setting, subjects were allowed to take home, assigned task, so that they can do the work with full dedication. No time limit was imposed to complete the task of OOA. Ratings for 12 parameters from 160 students were collected during the execution of the experiment. For what concerns the quality of data collection, we used pencil and paper; hence data collection could be considered critical. Finally the quantity and the quality of the data collected and its analysis were enough to support our conclusions. Parameters for measuring perceived usefulness and ease of use were taken from a published work in journal (Cockburn, 2000). In future, we will further improve reliability by increasing the number of questions. We accept this risk being a preliminary study and plan to replicate the experiment with more subjects in future.

Construct validity: Construct validity is the degree to which the independent variables and dependent variables accurately measure the concepts they purport to measure. We wanted to measure effect of Event-based approach in changing perception of users regarding ease of use and usefulness in using OOA technique. We also wanted to measure whether or not our Event-based would be easily understood by users. What will be their perception after learning new technique along with conventional approach? Dependant variables were measured by using questionnaire based on perceived ease of use and usefulness. We used 12 parameters which are objective measures that reflect perceived ease of use and usefulness of subjects (Cockburn, 2000). For this reason, we consider that they objectively measured what we purport to measure. We avoided the threat of a mono operation bias by providing the users with different types of tasks, deliverables and case studies that represent a significant range of software systems. We have no hypotheses guessing threat since the experiment was presented as a normal class exercise and the subjects were not informed of the hypotheses before the experiment.

Internal validity: Internal validity is the degree to which conclusions can be drawn about the causal effect of the independent variables. Internal validity judge whether observed changes can be attributed to a program or intervention (i.e., the cause) and not to other possible causes (sometimes described as “alternative explanations” for the outcome). We have avoided single group, multi group and social threats to internal validity by not forcing any subject to participate. It was a voluntarily involvement of all subjects chosen. Subjects were asked not to disclose their personal details. No bonus marks was allotted for the controlled experiment. Data collection and analysis was done by other faculty members. Subjects were not informed of the hypotheses before the experiment.

External validity: External validity is the degree to which the results of the research can be generalized to the population under study and other research settings. The greater the external validity, the more the results of an empirical study can be generalized with regards to actual software engineering practice. There was no bias selection in this experiment as users were randomly selected from two different courses and two different years of B. Tech Program. They were divided randomly into experimental and control group. Three threats to validity have been identified which limit the ability to apply any such generalization (a) the case studies used in the experiment are representative of real cases, but more empirical studies, using “real cases” from software companies, will be carried out (b) Although in this experiment students were used as subjects rather than professional practitioners, half of the sample was from B. Tech Final year of studies and close to their professional employment in industry. It is therefore reasonable to assume that if the experiment is done using professionals, the experiment should produce similar results. However this is a hypothesis that needs to be tested and could be the subject of a future work replication experiment. (c) This experiment was carried out towards the end of course delivery and it could be replicated in the mid phase of course delivery.

CONCLUSION

All Object-Oriented Analysis and Design (OOAD) methods start from the process of identifying objects and their classes from the requirements of the problem domain. But none of the methods to the best of our knowledge, have focused on event-based requirements analysis, rather all are behavioral based approaches. This study has described a systematic approach for requirement analysis of event-based systems. Requirements of the problem domain were captured as events in the proposed Event Templates. Mapping rules were applied to extract a domain model specification (analysis-level class diagram) from Event Templates. An Event-Meta Model has been proposed to focus on the concept of event as basis for class and object identification. The meta-model has addressed certain issues like what an event is, in the context of OOAD and why events should be the basis to derive static model of the system (class diagram). A comparative analysis is also done between Events and Use Cases Templates and it has been shown how our Event template is different from a conventional Use Case template and event tables used in other existing approaches.

A prototype tool 'EV-ClassGEN' has also been developed to provide automation support to extract events from requirements, document the extracted events in Event Templates and implement rules to derive specification for an analysis-level class diagram. The tool takes events occurring in the system as starting point in OOA and systematically derives an importable class diagram specification in XML Metadata Interchange (XMI) format for Argo UML tool. The proposed approach is also validated through a controlled experiment to compare the perceived ease of use and usefulness of the proposed event-based approach with a more conventional and industry standard Use Case based approach. Results of the controlled experiment have shown that after studying and applying Event-based approach, student's perception about ease of use and usefulness of OOA technique has significantly improved. Their project reports showed positive feedback about Event-based approach. These results reinforced the evidence that by analyzing events that are likely to happen in a system, one can derive class diagram information from requirements.

Our approach can well be applied to modeling real time systems, embedded systems and safety critical systems, where events play a significant role in understanding such domains. When applied to such domains, our approach can capture requirements in terms of domain events; model individual object's behavior and its collaboration and interaction with other objects in the domain.

The empirical study conducted in this study focused on users' perceptions, not on model quality or effort. In future, we plan to replicate the experiment for measuring the quality of class diagram and efforts used in generating class diagram using the two techniques. Our future work will also demonstrate how dynamic behavior of the system can be extracted from event templates. We are in a process of validating rules to transform the event templates to dynamic models. Additionally, we also plan to propose rules for generating test scenarios and derive some metrics from Event templates.

REFERENCES

Abbott, J.R., 1983. Program design by informal English descriptions. Commun. ACM., 26: 882-894. DOI: 10.1145/182.358441

Anda, B. and D.I.K. Sjberg, 2003. Applying use cases to design versus validate class diagrams-a controlled experiment using a professional modeling tool. Proceeding of the International Symposium on Empirical Software Engineering, Italy, Sept. 30-Oct. 1, IEEE Xplore Press, USA., pp: 50-60. DOI: 10.1109/ISESE.2003.1237964

Barber, K.S. and T.J. Graser, 2000. Tool support for systematic class identification in object-oriented software architectures. Proceeding 37th International Conference on Technology of Object-Oriented Languages and Systems, Nov. 20-23, Sydney, IEEE Xplore Press, NSW, Australia, pp: 82-93. DOI: 10.1109/TOOLS.2000.891360

Becker, L.B., C.E. Pereira, O.P. Dias, I.M. Teixeira and J.P. Teixeira, 2000. MOSYS: A methodology for automatic object identification from system specification. Proceeding 3rd IEEE International Symposium on Object-Oriented Real-Time Distributed Computing, Mar. 15-17, IEEE Xplore Press, Newport, CA., USA., pp: 198-201. DOI: 10.1109/ISORC.2000.839529

Booch, G., J. Rumbaugh and I. Jacobson, 2005. Unified Modeling Language User Guide. 2nd Edn., Addison-Wesley Professional, New York, ISBN: 10: 0-321-26797-4, pp: 496.

Cheong, C.W., 2008. Random walk models classifications: An empirical study for Malaysian stock indices. Am. J. Applied Sci., 5: 411-417. http://www.scipub.org/fulltext/ajas/ajas54411-417.pdf

Coad, P. and E. Yourdon, 1990. Object Oriented Analysis. 2nd Edn., Prentice Hall, Englewood Cliffs, NJ., ISBN: 10: 0136299814, pp: 233.

Cockburn, A., 2000. Writing Effective Use Cases. 2nd Edn., Addison-Wesley Professional, New York, ISBN: 10: 0201702258, pp: 304.

Davis, F.D., 1989. Perceived usefulness, perceived ease of use and user acceptance of information technology. MIS Q., 13: 319-340. http://www.jstor.org/stable/249008

Drake, J.M., W.W. Xie, W.T. Tsai and I.A. Zualkernan, 1993. Approach and case study of requirement analysis where end users take an active role. Proceeding of the International Conference Software Engineering, May 17-21, IEEE Xplore Press, Baltimore, MD., USA., pp: 177-186. DOI: 10.1109/ICSE.1993.346046

Dritsakis, N., 2004. A causal relationship between inflation and productivity: An empirical approach for Romania. Am. J. Applied Sci., 1: 121-128. DOI: 10.3844/ajassp.2004.121.128

Dritsaki, C. and A. Adamopoulos, 2005. A causal relationship and macroeconomic activity: Empirical results from European Union. Am. J. Applied Sci., 2: 504-507. DOI: 10.3844/ajassp.2005.504.507

Dritsakis, N. and K. Gialetaki, 2005. Macro-economic variables analysis in Ukraine: An empirical approach with cointegration method. Am. J. Applied Sci., 2: 836-842. DOI: 10.3844/ajassp.2005.836.842

Fang, Q. and Y. Liu, 2007. Empirical analysis: Business cycles and inward FDI in China. Am. J. Applied Sci. 4: 802-806. DOI: 10.3844/ajassp.2007.802.806

Ferg, S., 2003. What's wrong with use cases? Microsoft LPE. http://www.jacksonworkbench.co.uk/stevefergspages/papers/ferg--whats_wrong_with_use_cases.html

Jalloul, G., 2004. UML by Example. 1st Edn., Cambridge University Press, Cambridge, ISBN: 10: 0521008816, pp: 276.

Harmain, H.M. and R. Gaizauskas, 2003. CM-builder a natural language-based CASE tool for object-oriented analysis. J. Automat. Software Eng., 10: 157-181. DOI: 10.1023/A:1022916028950

Ilieva, M.G. and O. Ormandjieva, 2005. Automatic transition of natural language software requirements specification into formal presentation. Nat. Lang. Process. Inform. Syst., 3513: 392-397. DOI: 10.1007/11428817_45

Jacobson, I., G. Booch and J. Rumbaugh, 1999. Unified Software Development Process. 2nd Edn., Addison-Wesley Professional, New York, ISBN: 10: 0201571692, pp: 512.

Luckham, D., 2002. Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. 1st Edn., Addison-Wesley Professional, New York, ISBN: 10: 0201727897, pp: 400.

Kruchten, P., 2003. Rational Unified Process: An Introduction. 3rd Edn., Addison-Wesley Professional, New York, ISBN: 10: 0321197704, pp: 336.

Liang, Y., 2003. From use cases to classes: A way of building object model with UML. Inform. Software Technol., 45: 83-93. DOI: 10.1016/S0950-5849(02)00164-7

Liu, D., K. Subramaniam, B.H. Far and A. Eberlein, 2003. Automating transition from use-cases to class model. Proceeding of the Canadian Conference on Electrical and Computer Engineering, May 4-7, IEEE Xplore Press, USA., pp: 831-834. DOI: 10.1109/CCECE.2003.1226023

Liu, D., K. Subramaniam, A. Eberlein and B.H. Far, 2004. Natural language requirements analysis and class model generation using UCDA. Proceeding of 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, May 17-20, Springer, Ottawa, Canada, pp: 295-304. DOI: 10.1007/b97304

Muhairat, M.I., R.E. Al-Qutaish and A.A. Abdelqader, 2010. UML diagrams generator: A new case tool to construct the use-case and class diagrams from an event table. J. Comput. Sci., 6: 253-260. DOI: 10.3844/jcssp.2010.253.260

Mustafa, Y. and A. Awofala, 2004. Activity-based class design: an analytical method for deriving object-oriented classes. Iss. Inform. Syst., 5: 240-247. http://www.iacis.org/iis/2004_iis/PDFfiles/MustafaAwofala.pdf

Overmyer, S.P., L. Benoit and R. Owen, 2001. Conceptual modeling through linguistic analysis using LIDA. Proceeding of the 23rd International Conference on Software Engineering, May 12-19 IEEE Xplore Press, USA., pp: 401-410. DOI: 10.1109/ICSE.2001.919113

Perez-Gonzalez, H.G. and J.K. Kalita, 2002. GOOAL: A graphic object oriented analysis laboratory. Proceeding of the Companion of the 17th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Nov. 4-8, ACM Press, Seattle, Washington, pp: 38-39. DOI: 10.1145/985072.985092

Perez-Gonzalez, H.G., J.K. Kalita, A.S.N. Varela and R.S. Wiener, 2005. GOOAL: An educational object oriented analysis laboratory. Proceeding of the Companion to the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages and Applications, Oct. 16-20, ACM Press, San Diego, CA., USA., pp180-181. DOI: 10.1145/1094855.1094924

Poo, D.C.C., 1999. Events in use cases as a basis for identifying and specifying classes and business rules. Proceeding of the 29th International Conference on Technology of Object-Oriented Languages and Systems, June 4-7, IEEE Xplore Press, Nancy, France, pp: 204-213. 10.1109/TOOLS.1999.779013

Ross, R.G., 1988. Entity Modeling: Techniques and Applications. Database Research Group, Inc., USA., pp: 218.

Roussev, B., 2003. Generating OCL specifications and class diagrams from use cases: A Newtonian approach system sciences. Proceeding of the 36th Annual Hawaii International Conference on System Science, Jan. 6-9 IEEE Xplore Press, USA., pp: 1-10. DOI: 10.1109/HICSS.2003.1174886

Samarasinghe N. and S.S. Some, 2005. Generating a domain model from a use case model. Proceedings of the ISCA 14th International Conference on Intelligent and Adaptive Systems and Software Engineering, July 20-22, Natural Sciences and Engineering Research Council of Canada, Toronto, Canada, pp: 23-29.

Satzinger, J.W., B.R. Jackson and D.B. Stephen, 2006. Systems Analysis and Design in a Changing World. 4th Edn., Course Technology, USA., ISBN: 10: 1418836125, pp: 672.

Sharahili, Y. and Y. Liu, 2008. Empirical analysis II: Business cycles and inward FDI in China. Am. J. Applied Sci., 5: 1409-1414. DOI: 10.3844/ajassp.2008.1409.1414

Shlaer, S. and S.J. Mellor, 1988. Object Oriented Systems Analysis: Modeling the World in Data. 1st Edn., Prentice Hall, Englewood Cliffs, NJ., ISBN: 10: 013629023X, pp: 144.

Singh, S.K., S. Sabharwal and J.P. Gupta, 2009a. E-XTRACT: A tool for extraction, analysis and classification of events from textual requirements. Proceeding of International Conference on Advances in Recent Technologies in Communication and Computing, Oct. 27-28, IEEE Xplore Press, Kottayam, Kerala, pp: 306-308. DOI: 10.1109/ARTCom.2009.120

Singh, K., R. Gupta, S. Sangeeta and J.P. Gupta, 2009b. Automatic extraction of events from textual requirements specification. Proceeding of 2009 World Congress on Nature and Biologically Inspired Computing, Dec. 9-11, IEEE Xplore Press, Coimbatore, pp: 415-420. DOI: 10.1109/NABIC.2009.5393565

Some, S.S., 2005. Enhancement of a use cases based requirements engineering approach with scenarios. Proceedings of the 12th Asia-Pacific Software Engineering Conference, Dec. 15-17, IEEE Xplore Press, USA., pp: 25-32. DOI: 10.1109/APSEC.2005.64

Some, S.S., 2006. Supporting use case based requirements engineering. Inform. Software Technol., 48: 43-58. DOI: 10.1016/j.infsof.2005.02.006

Some, S.S., 2007a. Petri nets based formalization of textual use cases. SITE. http://www.site.uottawa.ca/eng/school/publications/techrep/2007/TR-2007-11.pdf

Some, S.S., 2007b. Use Case Editor (UCEd). Geeknet, Inc. http://sourceforge.net/projects/uced/

Song, I.Y., K. Yano, J. Trujillo and S. Luján-Mora, 2005. A Taxonomic Class Modeling Methodology for Object-Oriented Analysis. In: Information Modeling Methods and Methodologies, Krogstie, J., T. Halpin and K. Siau (Eds.). Idea Group, Hershey, PA., ISBN: 1591403766, pp: 216-240.

Turk, Z. and D.J. Vanier, 1993. Classification systems in object oriented modeling of buildings. Proceeding of the International conference Design to Manufacture in Modern Industry, June 7-9, NRC, Bled, Maribor, Slovenia, pp: 571-578. http://www.zturk.com/data/works/att/7f07.fullText.06493.pdf

Wahono, R.S. and B.H. Far, 2002. A framework for object identification and refinement process in object-oriented analysis and design. Proceeding of the 1st IEEE International Conference on Cognitive Informatics, (CI'02), IEEE Xplore Press, USA., pp: 351-360. DOI: 10.1109/COGINF.2002.1039317

Wiegers, K.E., 2005. More about Software Requirements: Thorny issues and Practical Advices. 1st Edn., Microsoft Press, USA., ISBN: 10: 0735622671, pp: 224.

Xu, C., S. Selvarathinam and W.X. Li, 2007. Sociopolitical instability and economic growth empirical evidence from Sri Lanka. Am. J. Applied Sci., 4: 1029-1035. DOI: 10.3844/ajassp.2007.1029.1035

Yourdon, E., 1988. Modern Structured Analysis. 2nd Edn., Prentice-Hall, New Delhi, India, ISBN: 10: 0135986249, pp: 688.