

**AN EVENT-BASED FRAMEWORK FOR OBJECT-ORIENTED
ANALYSIS, COMPUTATION OF METRICS AND
IDENTIFICATION OF TEST SCENARIOS**

*Synopsis of the Thesis to be submitted in fulfillment of the requirements for the
degree of*

DOCTOR OF PHILOSOPHY

By

SANDEEP KUMAR SINGH



Department of Computer Science Engineering and Information Technology

JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY

(Deemed University)

A-10, SECTOR-62, NOIDA, INDIA

January, 2011

AN EVENT-BASED FRAMEWORK FOR OBJECT-ORIENTED ANALYSIS, COMPUTATION OF METRICS AND IDENTIFICATION OF TEST SCENARIOS

Software is everywhere. It lets us get cash from an Automatic Teller Machine (ATM), make a phone call and drive our cars. An average company spends about 4 to 5 percent of its revenue on Information Technology (IT), whereas companies which are highly IT dependent, such as finance and telecommunications are spending more than 10 percent on it. In other words, IT sector has now one of the largest corporate expenses, outside employee costs. A lot of that money goes into hardware and software upgrades, software license fees, but a big chunk is for new software projects meant to create a better future for an organization and its customers.

Software projects are inherently complex, risky and require careful planning. Proper planning ensures that a project doesn't fail, while at the same time, customers get a clear definition of the project, know the project status and have a ready access to project deliverables at any point of time. Most recent surveys [1, 5] have shown that inadequate planning and specifications, ill defined requirements, poor process of requirement analysis and testing, lack of metrics and measures to compute project's sheer size and complexity, all together lead to numerous change requests, delays, significant added costs and increase in the possibility of errors. Thus a good requirement analysis method, proper management of software complexity and proper testing techniques are three important factors which play a vital role in avoiding software failures. This has been the motivation of the work carried out in this thesis. Next subsequent sections will focus on these three factors.

SOFTWARE REQUIREMENTS ANALYSIS

Understanding the problem domain and its requirements is a key to a successful project. Institute of Electrical and Electronics Engineers (IEEE) defines Requirements as a condition or a capability that must be met or possessed by a system, to satisfy a contract,

standard, specification or other formally imposed document. Somerville and Sawyer in [14] define requirements as a specification of what should be implemented.

Requirement engineering plays a vital role in understanding requirements. Requirement engineering process encompasses systematic and repetitive techniques in discovering, documenting, modeling and maintaining a set of requirements for a system as a whole, or specifically for software components of a system.

Findings on failures due to Poor Requirement Analysis

There are many studies recently done to quantify cost and causes of software failures [1, 5]. Statistics presented in articles [3] have shown that, **60% – 80% of project failures can be directly attributed to poor requirements gathering, analysis and management. In article [3], it is cited that 68% of IT projects fail primarily due to poor requirements.** In [5], it is projected that companies pay a premium of as much as 60% on time and budget, when they use poor requirements practices in their projects. Over 41% of the IT development budget for software, staff and external professional services is consumed by poor requirements at an average, where the company is using average analysts. Sloppy development practices are also a rich source of failure and they can cause errors at any stage of an IT project. Moreover, the costs of errors that are introduced during requirements phase and fixed later in the Software Development Life Cycle (SDLC) increase exponentially [6].

Conceptual Modeling for Requirement Analysis

Software process models produce conceptual system models after systematic analysis and documentation of requirements. These conceptual models are an important bridge between analysis and design process. Some of the popular conceptual modeling techniques are Data flow model (DFD), variants of DFD, Relational model, Entity–Relationship (ER) model, (Extended Entity–Relationship) EER model, E²R diagram, Higher-Order Entity Relationship Model (HERM), Semantic Database Model i.e. SDM, SOM (Semantic Object Model (SOM), Object Role Modeling (ORM), Conceptual Schema Language (CSL), DATAID-1 data schema integration, REMORA methodology, Booch method, Object-

Oriented Software Engineering (OOSE), Object-modeling technique (OMT) and Unified Modeling Language (UML).

Function-Oriented (F-O) methods [15] and Data-Oriented (D-O) methods [16] have paradigm mismatch between analysis and design, which is reduced by Object-Oriented (OO) software development methods like Booch method [17], OOSE or Jacobson method [18], OMT or Rumbaugh method [19], Coad and Yourdon method [21] and Wirfs-Brock method [22]. OO Conceptual Modeling provides a number of benefits like modularity, abstraction, information hiding and reusability that are not present in traditional requirement approaches and it uses the same model consistently from analysis to implementation. In the next section, we will discuss Object-Oriented Analysis (OOA) and some of its techniques and tools.

Object-Oriented Requirement Analysis

In OO Conceptual Modeling, a system is modeled as a group of interacting objects and is characterized by its class, its state (data elements) and its behavior. In 1997, Grady Booch, Jim Rumbaugh and Ivar Jacobson unified the Booch, OMT and OOSE method to develop UML which became an industry standard, created under the auspices of the Object Management Group (OMG) [20]. UML is a graphical language for visualizing, specifying and constructing artifacts of a software-intensive system and has become a de facto standard for OO conceptual modeling. In UML, Use case diagrams have been considered effective in modeling functional requirements of a system, in general and software component, in particular.

Techniques and Tools used for OOA

A vital decision during OO Conceptual Modeling is to find objects and classes during OOA and then build a conceptual model for a problem domain. In this section, we describe various techniques that have been used in the past to extract components from requirements for building class and object models. Techniques proposed have either used Natural Language Processing (NLP) or employed Use cases to identify classes. Some of these approaches have been automated by building prototype tools.

Techniques for OOA

Grady Booch popularized the concept of Russell and used singular nouns and nouns of direct reference to identify objects and plural and common nouns to identify classes [17]. However, this is an indirect approach to find objects or classes and all nouns are not always classes or objects. Some of them refer to entire assembly, subassembly, attribute or a service. Several parsers were built using this approach to extract nouns and noun phrases from large-scale texts.

In [23], authors have used computerized classification systems and thesauri for the purpose of OOA of a valid Slovenian earthquake code. This approach is also not very easy and straightforward and is burdened with checking candidate classes with a thesaurus and is very much dependent on the structure and completeness of a thesaurus. It is always difficult to find one-to-one mapping between a thesaurus term and a class. In 1991-1992, pioneers like Codd and Yourdan, Shaler and Mellor and Ross identified certain categories like persons, role and organizations etc, which define application domain entities. These categories help experienced analysts to identify classes or objects. This approach only finds tangible objects but fails to identify abstract classes [24]. Ed Seidewitz and Mike Stark developed a technique in [25] to identify objects, classes and services from terminators, data stores and data flows of DFD. This approach also suffers from several problems like use of data abstraction instead of object abstraction, scattered pieces of objects across several DFD's and fragmented objects and classes [25]. In [26], authors have presented an integration of several approaches under one head called taxonomic class modeling (TCM) methodology. This methodology is neither tested nor validated by a controlled experiment. It does not also provide any automation support. In [27], several approaches are presented, based on NLP of textual requirements to extract components of OOA model.

Work done in [28] has presented a Use case-driven development process and its validation. However, it is reported in empirical findings that this technique leads to problems, such as developers missing requirements and mistaking requirements for design. Work in [29] identifies classes from goals of each Use case without descriptions, instead of scenarios. In

[30], a set of artifacts and methodologies are used, to automate the transition from requirements to detail design. In [31], a process is proposed for generating formal OO specifications in Object constraint Language (OCL) and Class diagrams from a Use case model of a system through a clearly defined sequence of model transformations. Work in [32] presents a methodology and a CASE tool named Use-Case driven Development Assistant (UCDA) to automate natural language requirements analysis and class model generation based on Rational Unified Process (RUP).

Tools for OOA

Several authors have used techniques, described in the previous section, to develop automation support for analysts. Some of those tools are Use-Case driven Development Assistant (UCDA), MOSYS (A Methodology for Automatic Object Identification from System Specification), RARE (Reference Architecture Representation Environment), CM-Builder (Class Model Builder), Linguistic assistant for Domain Analysis (LIDA), OOExpert, AURA (Automated User Requirements Acquisition), GOOAL (A Graphic Object Oriented Analysis Laboratory). Work done in [33] presents automated approaches to extract elements of OO system (namely classes, attributes, methods and relationships between the classes, sequence of actions, the use-cases and actors) using NLP techniques.

Limitations of existing Object-Oriented Requirement Analysis Techniques

The techniques described in the previous section use long descriptive requirements, expressed in natural language, as a starting point. Apart from limitation of each of the approaches discussed earlier, a natural language description of requirements often has the problem of incompleteness, inaccuracy and ambiguity. On the other hand, Use Case approaches, although, quite popular are at a deviation from basic OO concepts of visualizing systems in terms of objects for object modeling. Moreover, recently several arguments against Use cases have been cited in literature [34].

Use Cases have document centric, time consuming and declarative nature. They have problem of invisible scope creep and have inability to differentiate dynamic and static

elements of specifications. Although, improvisation of Use case based requirements analysis approach has been done [35] 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 starting point still being Use Cases. In [36], it has been claimed that event modeling infuses rigor and discipline in Use case modeling by helping analysts in identifying what constitutes a Use case. According to [36, 37], event modeling helps in determining Use cases. Thus, one can say that Event modeling complements Use case modeling.

COMPLEXITY METRICS

Complexity is probably the most important attribute of software because it influences a number of other attributes such as maintainability, understandability, modifiability, testability and cost [7]. Basili defines complexity as a measure of the resources expended by a system, while interacting with a piece of software to perform a given task [8]. Work in [46] distinguishes three kinds of complexities: computational, psychological and representational, out of which, psychological complexity is the only complexity that is perceived by man. Structural complexity is a psychological complexity that has been studied extensively because it can be assessed objectively. Complexity is assessed by using metrics. Metrics measure the quality of a product, the productivity of people, and the benefits of new software tool or forms a baseline for various estimations.

Different metrics are proposed for different phases of software development. For instance, function points are used in requirements phase to estimate the size of the resulting system. Similarly, the metrics for cohesion and coupling are used for the design phase. The suite of metrics for OO design, Lines of Code, Software Science and Cyclomatic Number are helpful instruments for managing software process effectively. Other popular software metrics are Henry and Kafura 's Structural complexity, McClure 's Invocation complexity, Woodfield 's review complexity, Yau and Collofello 's stability measure, Yin and Winchester's architectural metrics based on analysis of a system's design structure chart and Douce's et al spatial complexity.

Findings on failures due to Complexity

In [4], it is reported that a project's sheer size is also a fountainhead of failure. Studies indicate that large-scale projects, fail three to five times more often, than small ones. *Greater complexity increases possibility of errors because no one really understands all the interacting parts of a whole or has an ability to test them.* Roger S. Pressman pointed out in his book, Software Engineering that "Even a small 100-line program with some nested paths and a single loop, executing less than twenty times, may require 10 to the power of 14 possible paths to be executed." To test all of those 100 trillion paths, he noted, assuming each could be evaluated in a millisecond, would take 3170 years. According to a report published in December 2009 [9], the primary cause of software project failures is complexity. Complexity can create delays, cost overruns and lead to inability in a system to meet business needs.

SOFTWARE TESTING

Testing is a process of detecting errors, injected into software during any phase of its development [10]. Testing has been found to consume at least half of the effort expended on total development [11].

Several approaches proposed for test case generation can be found in literature and these have been classified into random, path-oriented, goal-oriented and intelligent approaches [47]. Even though varied test case generation approaches are available, Model-Based Testing (MBT) approach has attracted many researchers and research is still being carried out to optimize the generation of test cases with minimum human effort. The next section presents the survey of related work done in the area of MBT using UML models.

Model Based Testing

A wide range of models like UML, SDL, Z, state diagrams, data flow diagrams, control flow diagrams, etc have been used in MBT. Some have used an EFSM (Extended Finite State Machine) or a state Machine (FSM). Others have used activity diagrams and I/O

explicit Activity Diagram (IOAD) model, sequence diagrams, State Diagrams and State Charts or an integrated approach using more than one specific UML model.

Findings on failures due to Poor Testing

In article [2], according to Gartner Research, “The lack of testing and Quality Assurance (QA) standards, as well as a lack of consistency, often lead to software failure and business disruption, which can be costly.” Results of a survey conducted in June 2010 [12] have also shown that majority of software bugs are attributed to poor testing procedures or infrastructure limitations, rather than design problems. A report that has cited six common failures of IT projects [13] has shown that poor testing and absence of proper change management are two main reasons of software failure.

TOWARDS AN EVENT-DRIVEN PARADIGM

Einstein, in his Theory of Relativity, used the term, ‘event’ to signify the fundamental entity of observed physical reality -- a single point in the time-space continuum. Similar to usage of events in modeling physical reality, the basic notion of event is also widely used to model software in different contexts and applications. Next, sub sections discuss the various existing applications and role of events.

Role of Events in Requirements Analysis & Specification, Modeling and Testing

Work in [38] has iterated the fact that event thinking is an equally important way of modeling system. This section describes some of the approaches that have used events for the purpose of requirements analysis and specification. One classical example is of Event partitioning approach described in [39]. Event partitioning approach has also been applied to Real -Time Systems Analysis to model ‘non-event’ events.

Events have been used to deliver OO architecture for systems of all sizes in, to blend Event and Object partitioning, to change state of objects, for identifying and defining requirements process and data patterns to capture the processing policy, to specify requirements using event calculus , to promote easier maintenance and implementation of

specifications for e-commerce application development, to model relevant facts and abstract behavior of application and to construct composite events from simple events.

Events have also played a very important role to model object interaction, develop a metric suite for domain modeling and for analogical reuse of structural and behavioral aspects of event-based object-oriented domain models. Events are modeled in terms of object-oriented structures, like entities and as four different events in UML language. Events are also used to model information structures and related activities in information systems and for modeling static and dynamic aspects of Information and Communication Systems.

Various event-based models have been used for GUI Testing [40] like an Event and Message driven Petri network (EMDPN) model and an EMDPN based interaction graph, Event InterActions Graph (EIAG). Events are also used to define a scalable and reusable model of GUIs, a GUI automation test model, a test coverage criterion for GUIs and for fault modeling.

Events in Complex Event Processing Systems

Event Processing is an emerging area. The academic roots of event processing originate in multiple disciplines: artificial intelligence, databases, simulation, verification, sensor handling, distributed computing, programming languages, business process management and many more. The EPTS (Event Processing Technical Society) [41] was launched in June 2008 as a consortium to promote the understanding of the event processing discipline, incubate standards and foster collaboration between industry and academia. An ACM conference, "Distributed Event-Based Systems" (DEBS) recognized as the "flagship" conference of the community, covers all topics relevant to event-based computing. Complex Event Processing (CEP) described in the book, "The Power of Events" [42] has shown how CEP can be used to enhance systems that deal with events. He showed how one can use the power of events to automate management without compromising managers' control. There are a few approaches that have used events for representing system requirements, identifying classes and generating class diagrams and use case diagrams.

Critical review of these approaches reveals that there are some drawbacks and a possible scope for improvement. This critical review is presented in the next section.

Techniques for OOA based on Events

Author in [43] has proposed a new alternative Event-Oriented modeling approach called Behavioral Pattern Analysis (BPA) for Safety Critical System, Railroad Crossing System, Production cell System and Multi-agent System. This approach has not focused on the issue of generating class diagram specification from requirements. It has rather proposed for improving class diagrams that are already generated from Use cases. Another approach [44] has used information from modified Event-Response Table described in [39] to build UML Class diagram and Use case diagram. The modified event table lacks to show trigger and causative relationship among events that can be very useful in building both static and dynamic models of a system. Moreover, the authors have not done any empirical validation of their approach. Neither have they formalized proper rules for generating class diagram specification. Authors in [37] have also used event partitioning approach to identify Use cases but have not focused on class diagram generation. Work done in [45] has used events in Use cases as the basis for identifying and specifying classes and business rules. This event-based approach, to the best of our knowledge has not received empirical validation of their process. Neither have they formalized any rules for generating class diagram specification nor do they have any automation support for the user.

PROBLEM STATEMENT

From the foregoing discussion, it is amply clear that inadequate planning and specifications, ill defined requirements, poor process of requirement analysis and testing, lack of metrics and measures to compute project's sheer size and complexity, all together lead to numerous change requests, delays, significant added costs and an increase in the possibility of errors. A good requirement analysis method, proper management of software complexity and proper testing techniques, play a vital role in avoiding software failures. From literature survey, following is concluded:

- (i) Natural language based approaches for OOA of requirements have their own inherent limitations. Use case based approaches for OOA have also received several critical reviews. Existing applications and perspectives of events have focused more on the aspect of processing of events. Though few approaches have been proposed to use events for OOA but these approaches either focus on improving Class diagrams that have been already generated from Use cases or generate Class diagrams from event tables without using proper rules. None of the existing event-based OOA approaches, to the best of our knowledge have addressed the issue of automatic extraction of events from requirements or have done any empirical validation of their process for OOA.
- (ii) None of the existing event-based OOA approaches, have used meta-modeling concepts to give justification of using events as a starting point in requirement analysis or have formalized any rules for generating class diagram specification and automating support for the user.
- (iii) None of the existing software metrics, to the best of our knowledge have used events and their interdependency as the basis for measuring complexity of a system.
- (iv) Existing Model-based testing approaches have extensively used UML models. Existing event-based test case generation approaches have considered GUI events only and have not dealt with other types of events like business events, state events or control events that happen in a system. Not much of an effort has been devoted to test the functional requirements of a system, more specifically for event-based systems.
- (v) Existing event-based test case generation approaches have used events either from source code, Petri net model, or from GUI applications, so they require the applications or code to be made available, prior to generating test cases. These applications are run on existing GUI automation framework to reverse engineer event-flow of an application. Not much of an effort has been devoted to use events captured directly from the requirements specification.

Therefore, objective of this thesis work is to propose:

An Event-based systematic approach to:

- 1 Develop an improvised process for OOA of requirements.

- 2 Generate standardized Class diagram specification based on XML Meta-data Interchange (XMI) standard.
- 3 Develop software metrics to measure complexity of a system based on system events at the analysis phase of SDLC.
- 4 Generate Test Scenarios for testing functional requirements of a system.
- 5 Provide automated support to reduce time and effort of analysts.

And hence the title:

*“An Event-Based Framework for Object-Oriented Analysis,
Computation of Metrics and Identification of Test Scenarios”*

ISSUES AND CONTRIBUTIONS

In order to propose an Event-based framework that is able to offer requirements analysis, design, modeling, program construction, testing and support activities of a typical software engineering process, following issues are identified and contributions are made to address the issues:

Issue 1: to identify, describe and document events from textual requirements.

Issue 2: to extract information to build conceptual models (class diagram specification) from requirements specifications for modeling the requirements.

Issue 3: to determine the complexity of a system at an analysis level, so that necessary changes can be made to reduce cost of developing such a system or avoid software failures, prior to design and implementation.

Issue 4: to generate test scenarios and test cases for testing functional requirements of a system.

Issue 5: to model changes in requirements and perform re-testing of the system after incorporating those changes.

Issue 6: to provide automated support for all issues described above, in order to reduce time and effort of analysts.

In order to address **issues 1 and 2** following contributions are made:

An Event-based framework is designed to integrate OOA, conceptual modeling, computation of system complexity and testing. The proposed framework, as shown in Figure 1, takes software requirements as input from which events are extracted either manually or automatically. The extraction process gives a list of events. Each event from the list is formally documented using Event Template. Event Templates are further used by Class Diagram Generator and Model Generator. Class Diagram Generator generates the XMI based Class Diagram Specifications whereas Model Generator generates an Event-Flow Model. The Event-Flow Model is fed to Test Scenario Generator and Metric Calculator that automatically generates Test Scenarios and computes the complexity of the system at analysis stage. An Event-Meta Model is also proposed for OOA to justify use of events as basis for class and object identification. The meta-model has addressed certain issues like what is an event, in the context of OOAD and why events should be basis to derive static model of a system (class diagram). The Event Meta-Model is shown in Figure 2.

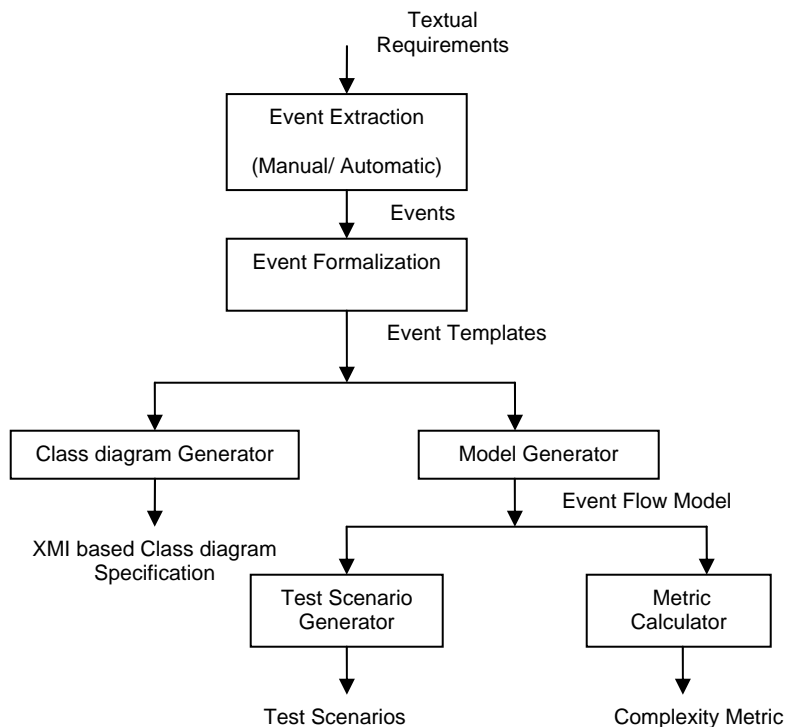


Figure 1: The Proposed Solution Approach

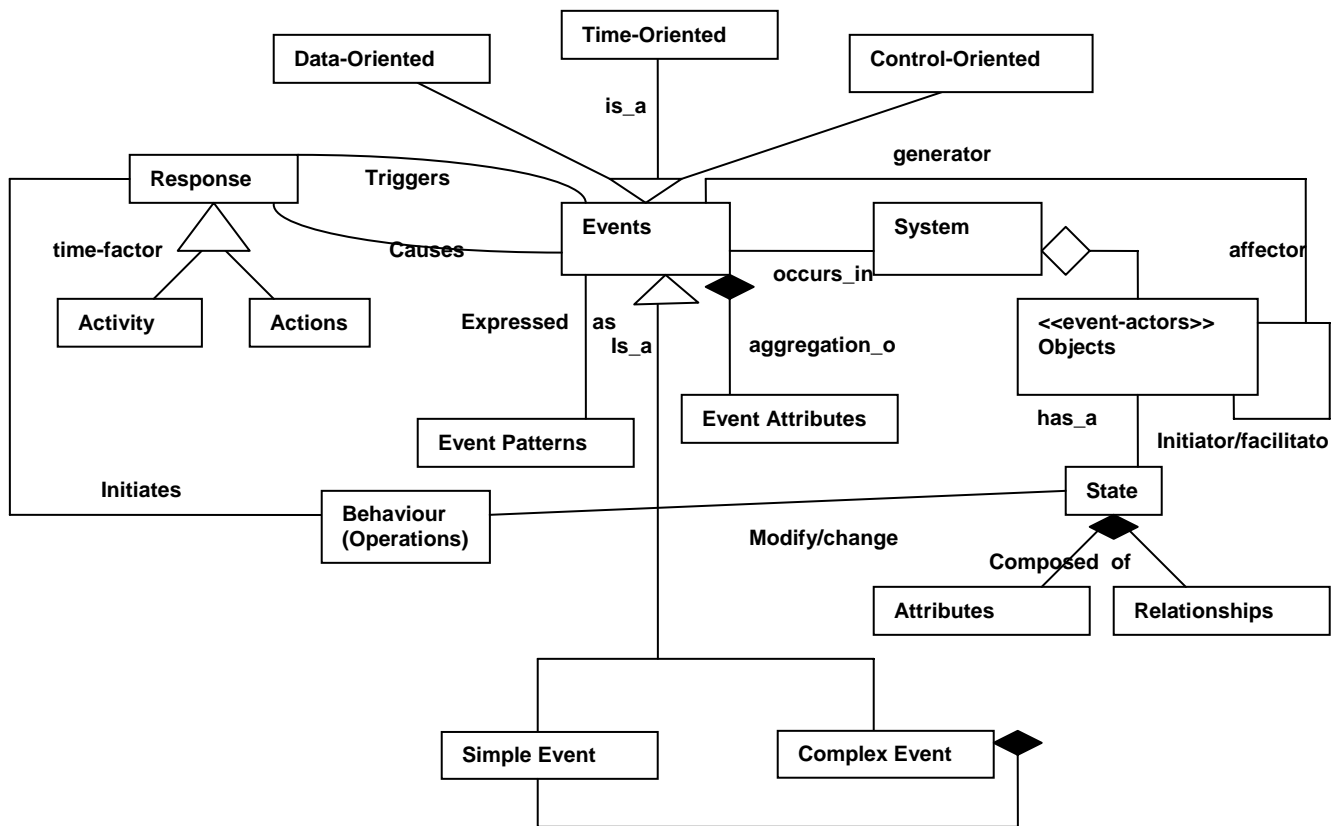


Figure 2. Event Meta-Model

A methodology is defined to derive standardized class diagram specification based on XML Meta-data Interchange (XMI) standard. For this, an Event Template, as shown in Table 1, is proposed to formalize and document events.

Table 1. Event Template for event “Sensor 1 generates detect signal at start place”

1	Event ID	EA07	
2	Event Name	Generates detect signal at start place	
3	Description	Sensor 1 generates detect signal at start place (State/Control Event)	
4	Initiator	Sensor 1	Count 1
5	Facilitator	ALCS / Belt 1(Start place)	Count 1
6	Affector	Signal	Count n
7	Timestamp	TA20	
8	Causative Events	EA05	
9	Inputs	Signal type	
10	Trigger Vector	Sensor 2 generates no-detect signal at scan place Sensor 3 generates no-detect signal at transition place Sensor 4 generates no-detect signal at end place	
11	Change-event	Connection between Sensor 1 and Signal	

A context free grammar is defined and based on this grammar an XML Schema is designed to validate the structure of an Event Template. A detailed comparison is also presented of an Event Template with a Use Case Template, with an aim to clarify the role and importance of Event Template in Event-based OOA. A set of 11 mapping rules viz. Class name rule, Role name rule, Class type (Boundary, Entity and Control) rule, Cardinality rule, Message passing rule, State rule, Creation rule, Association rule, Access rule, Modifier rule, Classify rule are defined to extract information for static model of a system from event templates.

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 subjects find an Event-based approach better in terms of perceived ease of use and usefulness of OOA technique on 10 out of 12 parameters.

The proposed methodology to generate class diagram specification is qualitatively compared with two existing approaches [44, 45]. In comparison with work in [44, 45], our methodology is better due to following reasons: (a) Events are directly identified and extracted from requirements (b) Unlike the structure of Event-Response Table and Event Script used in these existing approaches, structural content of event templates is designed to derive static model of a system. Temporal, causative and trigger relationship of events is exploited to derive static model of a system. (c) The process to automatically transform event templates to static model is formalized by developing mapping rules. (d) The output is standardized importable XMI format for Argo UML tool.

In order to address **issue 3** following contributions are made:

A matrix-based model comprising of two matrices: Causative events matrix (M_{cv}) and Trigger events matrix (M_{tv}), is proposed to develop metric suite consisting of Cumulative causative effect (C_{ce}), Cumulative trigger effect (C_{tv}), Complexity of an Event (C_e), Absolute complexity of a System (C_{systemabs}) and Relative complexity of a System (C_{systemrel}). These metrics compute system complexity from requirement specification

using event flows and event interdependencies. The proposed metric suite has also been evaluated in terms of Weyuker's properties, in order to guarantee that it qualifies as good and comprehensive. Results have shown that event-based metric to compute complexity fully satisfy eight of the Weyuker's nine properties, so it can be considered to have passed a significant part of the theoretical validation process. Therefore, it can be categorized as good, structured and comprehensive.

In order to address **issues 4 and 5** following contributions are made:

A methodology is proposed to generate event sequence-based test scenarios from Event-flow model. A Fault Model for the system under test (SUT) and test coverage criteria is defined for testing. An algorithm is also proposed to generate test scenarios from Event-Flow model and to detect faults described in the proposed Fault model. A methodology is proposed (a) to specify and model changes in software requirements, either in terms of addition and/or modification of events or event interdependencies (b) to apply a safe and efficient regression test selection technique on event-flow model, so as to select tests from test scenario that are deemed necessary to validate the modified software.

Finally in order to address **issue 6**:

A prototype tool is developed to implement the entire event-based framework. Tool Architecture as shown in Figure 3 has four core units - Event Extractor, Event Template Generator and Template Validator, Class Diagram Specification Generator and Complexity Metric Generator and Test Scenario Generator.

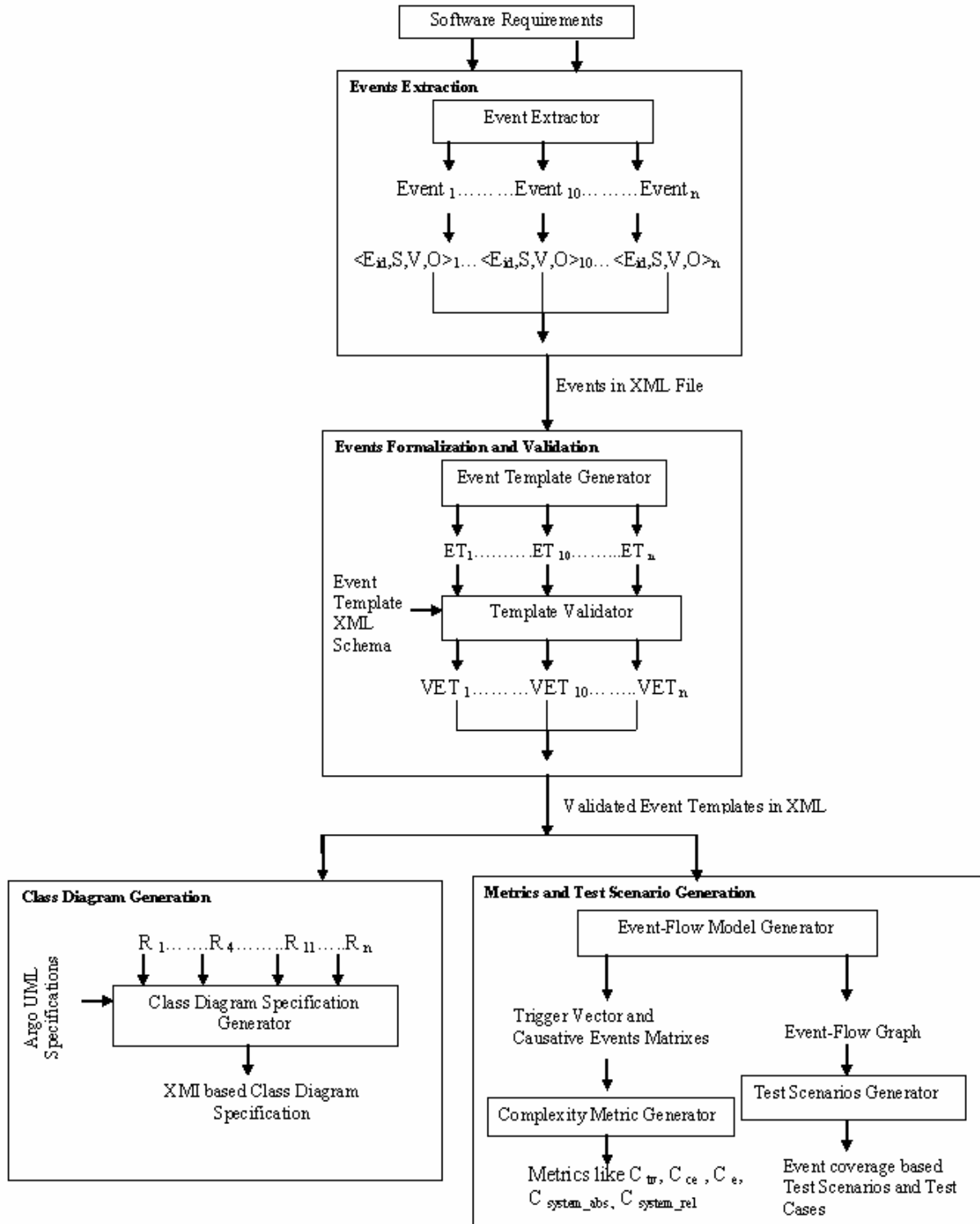


Figure 6.1: Tool Architecture of an Event-based Framework

ORGANIZATION OF THE THESIS

The Thesis is organized in seven chapters.

Chapter 1 This chapter presents an introduction and a detailed Literature review done to identify shortcomings and opportunities for the research work in the area of requirements analysis, modeling and testing. The objective of this chapter is to provide contextual background, identify gaps leading to the problem and discuss the approach proposed to solve the problem.

Chapter 2 presents the objective and principle behind an event based framework that forms the basis for defining the proposed Event Meta-Model. Structure of an Event Template for documenting events is also discussed. In the light of critical review of Use Case Approach, it also presents a detailed comparison of Event Template with Use Case Template.

Chapter 3 presents a detailed Event-based methodology and mapping rules for generating standardized Class diagram specification in XMI format, from Event Templates. This chapter also discusses a controlled experiment done to measure user's perception about perceived ease of use and usefulness of an event based approach in comparison with conventional Noun based and Use case based approaches for OOA.

Chapter 4 presents model based approach to construct software metric for analysis model. It first presents a detailed process to construct Event-Flow model based on interdependencies among events and then the metric is derived from Event flow model. The metric is termed as 'Event-Flow Complexity metric'. The proposed metric measures the complexity of system on the basis of event flows and their interdependencies. The metric has also been evaluated in terms of Weyuker's properties in order to guarantee that it qualifies as good and comprehensive.

Chapter 5 In this chapter, a model-based testing approach based on events is proposed for testing functional requirement of a system. The approach presents a process to derive test

scenarios and test cases from Event-flow model. A regression testing approach is also presented with objectives (a) to specify and model changes in software requirements in terms of addition and/or modification of events and (b) to apply a safe and efficient regression test selection technique on event-flow model, so as to select tests from test scenario that are deemed necessary to validate the modified software.

Chapter 6 describes the prototype tool that has implemented the entire event-based framework in five parts. First part deals with Event extraction. Second part deals with formalization of events. Third part deals with class diagram generation. Fourth part deals with computation of system complexity and the last part helps to automatically generate Test Scenarios and Test cases from a graph-based Event-Flow Model. The automation of the entire Event-based Framework reduces the time and complexity associated with each part.

Chapter -7 summarizes the work done in the thesis, highlighting the key contributions of the research work. It highlights all the areas where the proposed work can be successfully used. It also suggests scope for future work in this research. In the end, a list of all publications referred is given.

Keywords: Software Requirements Engineering, Event Patterns, Event Templates, Use Cases, Object-Oriented Analysis, Event Meta Model, Events, Automatic Event Extraction, Natural Language Processing, Dynamic Behaviour, UML, Document summarization, Event Flow Model, Testing, Event based coverage, Real Time systems, Software engineering, Complexity metrics, Event-based systems, Statistical technique, SVO Pattern, Event-Sequence based Test Case generation, Regression Testing, Event Processing.

REFERENCES

- [1] What makes software projects succeed? Available at <http://www.stylusinc.com/Common/Concerns/SoftwareProjectsFailure.php>
- [2] Dynamic Markets Limited 2007 Study of of 800 IT managers across eight countries :Two reasons Why IT Projects Continue To Fail Available at http://advice.cio.com/remi/two_reasons_why_it_projects_continue_to_fail
- [3] Cost of IT Failure Available at <http://www.objectwatch.com/whitepapers/ITComplexityWhitePaper.pdf>
- [4] Why Software Fails by Robert N. Charette // September 2005 Available at <http://spectrum.ieee.org/computing/software/why-software-fails>
- [5] Business Analysis Benchmark Available at http://www.iag.biz/images/resources/iag_business_analysis_benchmark_full_report.pdf
- [6] Boehm B.W., Verifying and Validating Software Requirements and Design Specifications, IEEE Software, vol. 1, no. 1, pp.75-88, 1984.
- [7] Brooks, F.P., Jr., No Silver Bullet: Essence and Accidents of Software Engineering, IEEE Computer, vol. 20, no. 4, pp.10-19, April, 1987.
- [8] Basili, V.R., “Quantitative Software Complexity Models: A Panel Summary” in Tutorials on Models and Methods for Software Management and Engineering, IEEE Computer Society Press, Los Alamitos, CA, pp.243-245, 1980.
- [9] The No 1 Cause of IT Failure :Complexity available at http://www.computerworld.com/s/article/345994/The_No_1_Cause_of_IT_Failure_Complexity
- [10] Bashir, I., “A Semi-exhaustive Testing Technique for C++ Classes Based on Intermember Relationship”, Ph.D. Thesis, Syracuse University, December 1993.
- [11] Edward, K., “Software Testing in the Real World: Improving the Process”, 1st ed., NY: Addison-Wesley / ACM Press Book, pp.272, 1995.
- [12] Off-shoring: A List of CMM-5 Certified Companies in India available at http://www.kamat.com/indica/current_affairs/outsourcing/cmm5_list.htm
- [13] Why Projects Fail available at <http://www.coleyconsulting.co.uk/failure.htm>
- [14] Sommerville, Ian and Sawyer, Pete, ”Requirements Engineering: A Good Practice Guide”, 1st ed., New York, NY, USA : John Wiley & Sons, Inc.,pp. 404, 1997 .
- [15] Demarco, T. “Structured Analysis and System Specification”, Facsimile ed., Prentice Hall,

pp.352, 1979.

- [16] Chen, P. "Entity-relationship Approach to Data Modeling", ACM Transactions on Database Systems, vol.1, no. 1, pp.9-36, 1976.
- [17] Booch, G., "Object-Oriented Development", IEEE Trans. Software Eng, vol. 12, no. 2, pp. 211-221, Feb.,1986.
- [18] Jacobson, I., Christerson, M., Jonsson, P., Overgaard, G., "Object-Oriented Software Engineering", 1st ed., Addison-Wesley, pp-552, 1992.
- [19] Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., "Object-Oriented Modeling and Design", 1st ed., Prentice-Hall, pp.512,1991.
- [20] UML Specification available at <http://www.omg.org/spec/UML/2.0/>
- [21] Coad, P. & Yourdon, E., "Object-Oriented Analysis", 1st ed. Prentice Hall, Englewood Cliffs, N.J., pp.200, 1989.
- [22] Wirfs-Brock, R., Wilkerson, B. & Weiner, L., "Designing Object-Oriented Software", 1st Int. ed., Prentice-Hall, pp.341, 1990.
- [23] Turk, Ž. and Vanier, D., "Classification Systems in Object Oriented Modeling of Buildings", in Proc. of Int. Conf. on Design to manufacture in Modern Industry, Maribor, Slovenia, pp. 571-578, 1993.
- [24] Coad, Peter and Yourdon, Edward, "Object-oriented Analysis", 2nd ed.NJ: Prentice Hall, Englewood Cliffs, pp.233, 1990.
- [25] Seidewitz, Ed.and Stark, Mike, "Toward a general object-oriented software development methodology", ACM SIGAda Ada Letters, vol.7, no.4, pp.54-67, July, 1987.
- [26] Krostige, J., Halpin, T. and Siau, K., "A taxonomic class modeling methodology for object oriented analysis", Information Modeling Methods and Methodologies: Advanced Topics in Database Research, pp. 216-240, 2004.
- [27] Saeki, Motoshi,Horai, Hisayuki and Enomoto, Hajime,"Software development process from natural language specification", in Proc.of the 11th Int. Conf. on Software engineering, Pittsburgh, Pennsylvania, United States, pp.64 - 73, 1998.
- [28] Bente Anda, Dag I. K. Sjøberg., "Investigating the Role of Use cases in the Construction of Class diagrams", Journal of Empirical Software Engineering, vol.10, no.3, pp. 285-309, July,2005.
- [29] Liang, Ying.,"From Use cases to classes: a way of building object model with UML", Information & Software Technology, vol. 45, no. 2, pp.83–93, Nov,2003.

- [30] Liu, D. Subramaniam, K. Far, B.H. Eberlein, A., “Automating transition from use-cases to class model”, in Proc. of IEEE Canadian Conference of Electrical and Computer Engineering, Canada, vol. 2, pp.831- 834, 2003.
- [31] Roussev, B., “Generating OCL specifications and Class diagrams from Use cases: a Newtonian approach”, in Proc. of the 36th Annual Int. Conf. of System Sciences, Hawaii, pp. 321-331, 2003.
- [32] Dong Liu, Kalaivani Subramaniam, Armin Eberlein, and Behrouz H. Far., “Natural Language Requirements Analysis and Class Model Generation Using UCDA”, in Proc. of 17th Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2004, Ottawa, Canada, pp.295–304, 2004.
- [33] Nanduri, S. and Rugaber, S.,”Requirements validation via automated natural language parsing” , Journal of Management Information Systems, vol.12 , no.3, pp. 9 – 19, Jan ,1995.
- [34] Arlow, Jim, “Use cases, UML Visual Modelling and the Trivialisation of Business Requirements”, Requirements Eng, vol.2, no.3, pp. 150-152, June, 1998.
- [35] Some, S.S., “Enhancement of a Use cases based requirements engineering approach with scenarios”, in Proc. of 12th Asia-Pacific Software Engineering Conference, Taiwan, pp. 25-32, 2005.
- [36] Use Cases that Work : Using Event Modeling to infuse rigor and discipline into Use Case Analysis , available at www.ocgworld.com/doc/OCG_Use_Cases_that_Work.pdf
- [37] Satzinger John .W, Jackson Robert .B., Burd Stephen D, “Systems Analysis and Design in a Changing World” 5th ed.: Course Technology, pp. 783, 2008.
- [38] Bolognesi, Tommaso, “On state-oriented vs. event-oriented thinking in formal behavioral specifications”, Tech. Rep. 2003-TR-20, Sep. 2003.
- [39] McMenamin, Stephen M. and Palmer, John F. “Essential Systems Analysis” 2nd ed. USA: Prentice-Hall Yourdon Press, pp.408, 1984.
- [40] Jun-yi Li, Hong-fang Gong, Ji-ping Hu, Bei-ji Zou and Jia-guang Sun., “Class Hierarchical Test Case Generation Algorithm Based on Expanded EMDPN Model”, Journal of Central South University of Technology, vol. 13, no. 6, pp. 717-721, Dec.2006.
- [41] Event Processing Technical Society available at <http://www.ep-ts.com/>
- [42] Luckham, David C., “The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems”, 2nd ed., Boston, MA : Addison-Wesley Longman Publishing Co. Inc., pp.400, 2001.
- [43] El-Ansary A. “Behavioral Pattern Analysis: towards a new representation of systems requirements based on actions and events,” in Proceedings of the 2002 ACM Symp.on Applied

computing, Madrid, Spain, pp.984 – 991, 2002.

- [44] Mohammad I. Muhairat, Rafa E. Al-Qutaish and Akram A. Abdelqader, “UML Diagrams Generator: A New CASE Tool to Construct the Use-Case and Class diagrams from an Event Table” *Journal of Computer Science*, vol. 6, no. 3, pp. 253-260, Mar. 2010
- [45] Danny C.C.Poo, “Events in Use cases as a Basis for Identifying and Specifying Classes and Business Rules” in *Proc. 29th Int. Conf. on Technology of Object- Oriented Languages and Systems*, France, pp.204-213, 1999.
- [46] Fenton, E. Norman, “Software Metrics: A Rigorous Approach”, 2nd ed. UK: Chapman & Hall, pp 320, 1991.
- [47] Prasanna, M., Sivanandam, S.N., Venkatesan, R. & Sundarrajan, R., “Survey on Automatic Test Case Generation”, *Academic Open Internet Journal*, available at <http://www.acadjournal.com/2005/v15/part6/p4/>.

LIST OF AUTHOR'S PUBLICATIONS

1. K.Singh, Sandeep and Sabharwal, Sangeeta and Gupta, J.P., "Object Oriented Analysis using Event Patterns" in Proceedings of International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CIS2E 07) December 3 - 12, 2007,Page 438-442.
(www.springerlink.com/index/nm01023027574507.pdf).
2. K.Singh, Sandeep and Sabharwal, Sangeeta and Gupta, J.P., "Event Patterns for Object Oriented Requirement Analysis" in Proceedings of IASTED International Conferences on Advances in Computer Science and Technology, April 2 - 4 , 2008, Page 115-120 (**Indexed in Scopus, ACM**).
3. K.Singh, Sandeep and Sabharwal, Sangeeta and Gupta, J.P., "Events-An Alternative to Use Case as starting point in Object-Oriented Analysis", Proceedings of the 2009 Second International Conference on Emerging Trends in Engineering & Technology, Pages: 1004-1010 , Year of Publication: 2009 ISBN:978-0-7695-3884-6 . (**IEEE, ACM, DBLP indexed**).
4. K.Singh, Sandeep and Sabharwal, Sangeeta and Gupta, J.P., "E-XTRACT: A Tool for Extraction, Analysis and Classification of Events from Textual Requirements", Proceedings of the 2009 International Conference on Advances in Recent Technologies in Communication and Computing, Pages: 306-308, Year of Publication: 2009, ISBN:978-0-7695-3845-7 . (**IEEE, ACM, DBLP indexed**)
5. K.Singh, Sandeep and Sabharwal, Sangeeta and Gupta, J.P., "Event-based Metric for Computing System Complexity" In Communications in Computer and Information Science, 2010, Volume 95, Part 1, 46-61.
(**Indexed in ISI Proceedings and Scopus, Published by Springer**).
6. K.Singh, Sandeep and Sabharwal, Sangeeta and Gupta, J.P., "**An Event-based Methodology to Generate Class Diagrams and its Empirical Evaluation** ". Journal of Computer Science, Science Publications, USA, ISSN: 1549-3636,

Abstracting/Indexing : Ulrich , DOAJ , Cabell , WAD , ASA , IET- ISI Thomson Scientific , IET , Genamics , EBSCO , Thomson Gale , ProQuest , SCOPUS , Current Abstracts , Zeitschriften , Expanded Academic ASAP , InfoTrac Custom , Science Resource Center , Student Resource Center College.
7. Sabharwal, Sangeeta and K.Singh, Sandeep and Gupta, J.P., "**Deriving System Complexity Metric from Events and its Validation** ".International Journal of Software Engineering and Knowledge Engineering (IJSEKE), Submitted on 8th July 2010, Received Acceptance with minor revision on 4th November 2010, Resubmitted with revisions on 19th November 2010.

Abstracting/Indexing : SciSearch® ,ISI Alerting Services ,CompuMath Citation Index® (CMCI®) ,INSPEC ,DBLP ,Bibliography Server ,io-port.net ,Compendex ,Computer Abstracts ,Scopus.

Online Gateways : CrossRef ,SwetsWise ,EBSCOhost Electronic Journals Service ,CNPIEC ,EBSCO Publishing ,OCLC ,FirstSearch ,WorldSciNet - Mirror Site in China ,Google Scholar ,J-Gate ,CEPIEC .

8. Sabharwal, Sangeeta and K.Singh, Sandeep and Gupta, J.P., “**A Novel Approach to Derive Test Scenarios from Event Flow Model**”. Submitted for review on 8th July 2010 to Int .Journal of Software Testing, Verifiability and Reliability. Current Status: Under First Revision.

Abstracting and Indexing Information

Web of Science (Thomson ISI), SCOPUS (Elsevier), INSPEC (IET), COMPENDEX (Elsevier), Computing Reviews (ACM), Cambridge Scientific Abstracts (CSA/CIG), Computer Science Index (EBSCO) etc.