International Journal of Computer Science Engineering Research and Development (IJCSERD), ISSN 2248 – 9363(Print)
ISSN 2248 – 9371(Online), Volume 3, Number 1
Jan- March (2013), pp: 12-22

IJCSERD

© PRJ PUBLICATION

EVENT DRIVEN OPTIMIZER: AN APPROACH TO OPTIMIZE THE EVENTS THAT TRANSITION FROM INCEPTION TO ELABORATION PHASE

Om Kumar C.U[1] Phalguna Krishna E.S.[2] Chaitanya Varma.M[3]

[1]Sree Vidyanikethan Engineering College -Dept of CSE- Tirupathi,A.P
[2]Asst Prof, Sree Vidyanikethan Engineering College -Dept of CSE- Tirupathi,A.P,
[3]Sree Vidyanikethan Engineering College -Dept of CSE- Tirupathi, A.P

ABSTRACT:

Research indicates that a week's effort on coding and testing is equal to 9 weeks' effort on its requirements gathering. Most of the companies today put their effort initially on project planning. They follow a strict determined well-planned approach to meet their deadlines. Researches on Requirements Gathering encompass a wide range of techniques. It is clear from the research that a System can be developed at ease with those techniques but those techniques do not help Users' in specifying that System. This paper discusses Event Driven Approach that helps in specifying System by gathers Requirements in less time. By using Stochastic Event Parser we classify the listed Events and by applying Monte Carlo Event Correlator we find Correlated Events. We provide a prototype called Event-Driven Optimizer that optimizes the Correlated Events thus producing minimized, Unique and Functional Events.

Keywords: Event, Event Aggregator, Stochastic Event Parser, Monte Carlo Event Correlator, Event Optimizer.

I INTRODUCTION

Most of the companies in this Digital age plan ahead of time to satisfy their Clients. They do so, by following a strict determined approach so that they meet their deadlines. Some of the techniques that are used to gather Requirements to satisfy the Clients are discussed in section 2. Although the techniques are said to be different they came into existence because of one property. That is nothing but Agile property. Almost all the project Requirements are Agile (i.e)

Requirements tend to evolve incrementally. When Requirements tend to evolve in an incremental fashion then it concludes that the initial Requirements are not concrete. When a projects Requirements are not concrete then what is the use of spending one fourth of its life time in Requirements gathering. To overcome this drawback a technique has to be developed that gathers Requirements in less time when compared to the other listed techniques. One such approach is established in this paper. We derive requirements using Event-Driven Approach. This technique uses Event Table to record the Events. An Event is an incident that is inconsistent, with the ordinary course or expected outcomes.

Events in general are very diverse such as

I. External
II. Internal and
III. Conditional Events.

The Event table records all the Events including Source and Destination of those Events. With this Event table one can derive use case diagrams. The Source and Destination attributes in the Event table turns up into Actors in the Use case Diagrams. The action occurred is recorded as Event attribute in the Event Table. This Event attribute can be used to derive Use Cases. With relationships like Generalization, Association, Dependency and Realization are used between the Event and actors, a whole Use case Diagram can be derived from the Event Table.

This Event-Driven Approach gathers Functional Requirements in less time when compared to the other listed techniques in section 2. Even here Requirements are tend to Agile but since less time is invested in gathering those Agile Requirements by opting an Incremental strategy, it wouldn't consume much time and effort. Here the whole approach depends on Event Table. More care is to be taken when Events are recorded in the Event table as they finally play a vital role in project phase transition. Event table helps in transition of the project from Requirements phase through Requirements Gathering to Design phase by deriving Use case diagrams from the gathered Requirements. The other main challenge that is to be overcome is the limit towards the total number of Events. Since the Requirements' Gathering happens manually everything that a client and a developer considers necessary gets recorded as an Event in the Event Table. This paper proposes a technique that takes Event Table as an input and produces Optimized events that are unique and functional. We Optimize the Events by using

I. Event Table
II. Event Aggregator
III. Stochastic Event Parser
IV. Monte Carlo Event Correlator and
V. Event Optimizer

Related Work

Generally Requirements play a vital role in Project Development. They are gathered to lay down the actual Requirements of the Customer. The different Requirements' Gathering Techniques are explained below.

Table I

Types of Requirements Gathering Techniques

S.No	Technique	Description
1.	One on One Interview	A session planned ahead of time, wherein the requirements gatherer fires set of questions to uncover requirements [1].
2.	Group Interviews	A technique wherein a group of stakeholders are interviewed to extract a rich set of requirements in a short period of time. They require more planning since more than one stakeholder is involved [2].
3.	Facilitated Sessions	In this technique a team based approach is used to extract high quality information within the prescribed time frame [2].
4.	Joint Application Development	This was initially developed by chuck morris & tiny Crawford of IBM to bring system developers and users of varying backgrounds and opinions together in a productive and creative environment [2] [3] [10].
5.	Prototyping	A visual model built as an initial version of solution which can be cycled around with clients intervention [4].
6.	Use cases	These are a list of steps that occurs in a case of the use of a system. They describe the interactions between actor and a system [4].
7.	Following People Around	A technique that involves catching up with clients. They are used when you need to get Hands-on feel of how business function works today [5].
8.	Request for Proposals (RFP's)	This technique is used by a Vendor where he receives Requirement's as Request For Proposals'. The Vendor decides to take up the project or not by matching the Requirements with his capabilities [6][7].
9.	Brainstorming	When a solution is to be created by a set of ideas that people agree to, this technique is used. Experts sit in a room create and communicate ideas which are finally prioritized by stakeholders [1] [7].
10.	Document Analysis	A technique which analyzes the given document.
11.	Reverse Engineering	A technique that discovers the technological principles behind a device, object or a system by analysing its functional structure and operation [1] [8].
12.	Surveying	A method involving a list of questions that uncovers stakeholder's requirements or needs [3].
13.	Requirements Workshop	A technique used to discover requirements in a quick and better fashion. Here Right people are brought under one roof and by getting them correct the quality of the requirement document is improved [1].
14.	Storyboards	This technique is borrowed from film industry to generate and explore alternatives to test the feasibility of a specific approach [4].
15.	Wireframe	This technique is a bare user interface design that shows how it will be constructed without aesthetics. It allows the user to get a visual of his system at minimal cost [2].
16.	Imprint analysis	This technique refers to a collection of associations and emotions unconsciously linked to a word, concept or experience. A Requirement is given as keyword to client and based on his emotions a decision is concluded [9] [10].

From the above listed Requirements gathering technique it is clear that the Requirements are Agile. To cope up with the Agile Requirements different strategies have evolved. Even though many techniques have evolved there is still a gap between client requirement and developers' deliverable. It is only due to that gap, the Developer is not able to extract what the client actually requires. When the requirements are not concrete, agile why should the Requirements Manager spend one fourth of the project time in Requirements gathering.

Event- Driven Approach

Event Table:
This table records' all the Events which can be either external, internal and Conditional Events. The Event table records these events as Event attribute.

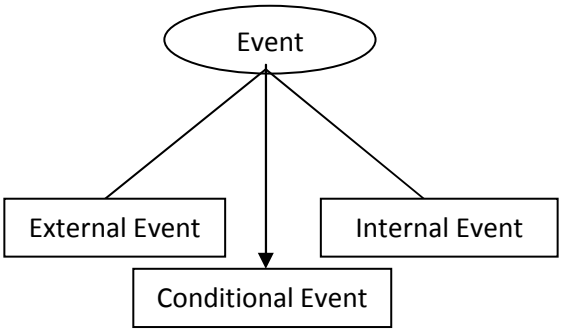

Figure: 1 Type of Events

Apart from this the table also records the source that triggered the Event and the Destination of the triggered Event as Source and Destination attributes. The formal Event is recorded using Action attribute in the table. The table now gives complete information of the clients' necessity or needs of the client to the developer. Now this can be said as a Requirements gathering technique [11].

Table II
Requirements Gathering Event Table

Source	Event Triggered	Action	Event	Destination
client	External	Processing Loan	Request for a Loan	Loan manager
Customer	External	Place Order	Ordering of items	Shopping cart
System	Internal	Validate User	Authentication of user	User
Student	Conditional	Request for book	Request for issual of a book in his account	Librarian

From the above table Functional requirements can be derived. The gathered Requirements are nothing but Events [5].

The Requirements gathered are as follows.

1. Processing Loan.
2. Place Order.
3. Validate Order
4. Request for Book

Transition from Requirements Phase to Design Phase

When relationships are added to the Event Table the table gets transitioned from Requirements gathering phase to Design Phase. We say it design phase because once relationships are added in the Event table it is easy to derive Use case diagrams from it. The actual Action becomes the Use case, Source and Destination attributes turns in to Actors [11]. When Relationships are applied between Actors and Use cases we come up with a Use case diagram thus causing a transition from Requirements Gathering to Design Phase [11].

Table III Use Cases being derived from Event Table

Source	Event Triggered	Usecase	Relationship	Event	Destination
client	External	Processing Loan	Association	Request for a Loan	Loan manager
Customer	External	Place Order	Association	Ordering of items	Shopping cart
System	Internal	Validate User	Association	Authentication of user	User
Librarian	Conditional	Manage Customer Account	Aggregation	Updates customer information	Library DB

The Use case diagram derived is as follows [12].

1)

Figure: 2 External Event

2)

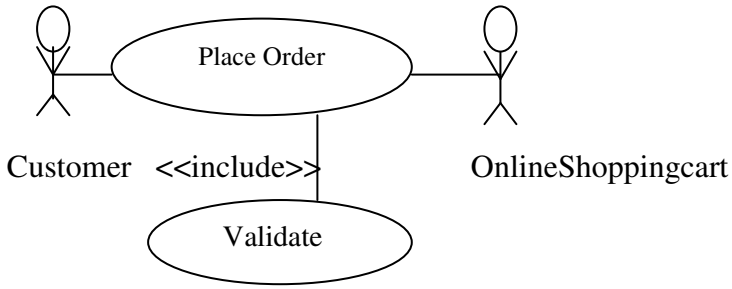

Figure: 3 Internal Event

3)

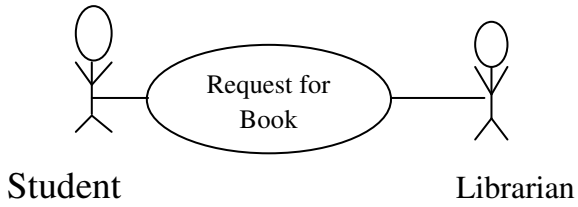

Figure: 4 Conditional Event

As shown above any number of events can be generated. They can be generated as Requirements or can be use cases [12] [13]. The main challenge here is to limit the number of Events. Practically limiting of Events to a particular number would decline the quality, since it does not allow all the Events to occur which in turn during Transition does not derive all the Use cases [14] [15]. A practical solution to this challenge would be to allow 'N' number of Events to occur and then optimize it. The following architecture describes the Event-Driven Optimization process in detail [16].

Event Optimization

This proposed architecture allows 'N' number of Events and Optimizes the recorded Events finally [17]. Individual Events are recorded separately in the Event Table in the format shown in Table III. All the Events recorded are aggregated by using Event Aggregator. The aggregated events are parsed by applying prioritization technique. After prioritizing the Events, the Monte carlo Correlator finds the Correlated Events. Manually the Correlated Events are checked and are replaced by a single Event thus optimizing the total number of Events without declining the quality.

Architecture:

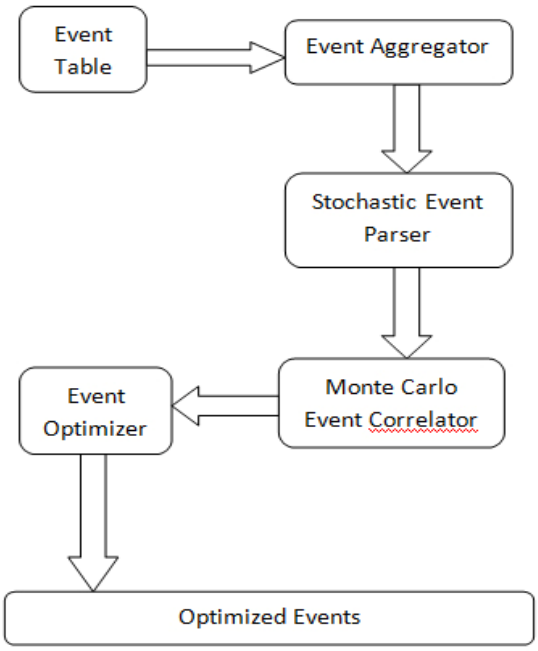

Figure: 5 Event Optimization

Event Table:

Every Event (External, Internal, and Conditional) along with the Source (where the Event was initiated) and Destination (Where it ends up) are represented using relationships in Event Table [11]. The format is shown in Table III.

Event Aggregator:

Each and every event along with its complete details is given as input to Event aggregator. All the Events are stored in its Database.

Stochastic Event Parser:

This maintains a set of rules that classifies the different Events into different kinds [18]. The Source, Destination and Event attributes in Event Aggregator are statically parsed using Stochastic Event Parser. Each and every Event from the Parser is mapped with one another to find the similarity among them [18] [19]. Priorities are used here to map the related Events.
Events with same Source and Destination are given medium priorities. Events with different source and Destination are given low Priorities.

Monte Carlo Event Correlator:

This uses numbers as a problem solving technique to find correlated Events and classifies Events from Stochastic Event Parser into 3 categories

1. High
2. Medium, and
3. Low prioritized Events.

The Monte Carlo Event Correlator considers the medium prioritized events of Stochastic Event Correlator. 2 or more Events with common Actors are said to correlated Events if they have a related Action attribute. Those Events are said to be correlated by Monte Carlo Event Correlator. For Readers understand ability we have used colors to represent correlated Events. Simply if 2 Events with medium priority have related action then the module provides high priorities to it.

Event Optimizer:

The Event optimizer performs Event Optimization manually by filtering related Events. High Prioritized Events are nothing but related Events. They are optimized by replacing it with a single Generalized Event. Thus all the Events now in the Event Optimizer are minimized, unique and functional.

Experimental Analysis

We have implemented a prototype of our proposed system using Java. In our prototype, we have implemented Event Aggregator, Stochastic Event Parser, Monte Carlo Event Correlator. We have tested the consistency of the system and have monitored its performance.

The final Optimization is achieved only through an human intervention. Simple Event Correlator (SEC) a tool for accomplishing event correlation tasks in the domains of log analysis, system monitoring, network and security management is considered for developing this particular tool [20]. Since SEC uses Perl for its execution and are mostly used in networking domain, we have used the logic behind it to create this Event-Driven Optimization tool.

The Event Table Stores every single Event as shown in Table III. To get an overview on all the Events recorded, Event Aggregator is used. All the individual Events are aggregated by using this module. A sample Event Aggregator containing 15 Events are shown in fig 6. The Syntax of Event Aggregator is similar to Event Table since they aggregate ‘N’ Events in the Event Table.

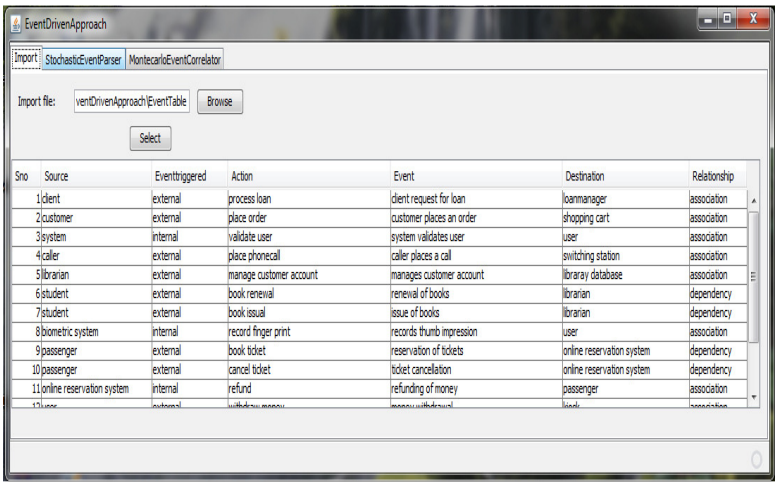

Figure: 6 Event Aggregator

The imported table is passed as input to the Stochastic Event Parser. Here Events are classified by Parser Rules. Prioritization is considered for categorizing the 'N' Events imported from Event Aggregator. Priorities applied on Events are of two types.

1. Low prioritized Events and
2. Medium prioritized Events.

An Event is said to be low prioritized if it's Source and Destination (Actors) doesn't match with any of the other listed Events. Simply Low prioritized Events are Events that have Unique Actors.

An Event is said to be medium prioritized if it's Source and Destination (Actors) matches some other Events. Simply When 2 Events have Actors in Common they are medium prioritized. Fig. 7 gives you a list of Events based on Prioritization.

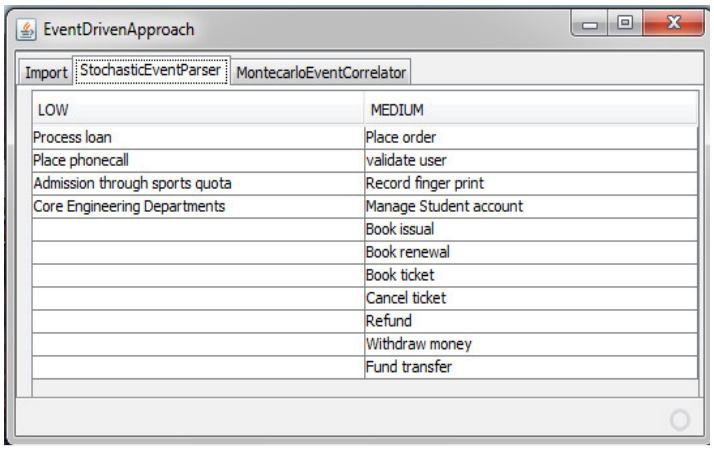

Figure: 7 Stochastic Event Parser

Monte Carlo Event Corrrelator optimizes the medium prioritized Events. As mentioned earlier Medium prioritized Events are those who have Actors in common. To optimize them again here

those Events are checked with their Action attribute. If some medium prioritized Events have their Action in common then Monte Carlo Event Correlator considers those Events as. For example in Fig 8 the first3 Events list in Medium Column are said to be correlated, the next 3 again are said to be correlated among themselves. For Reader's understandability we have represents the Correlated Events with similar colours.

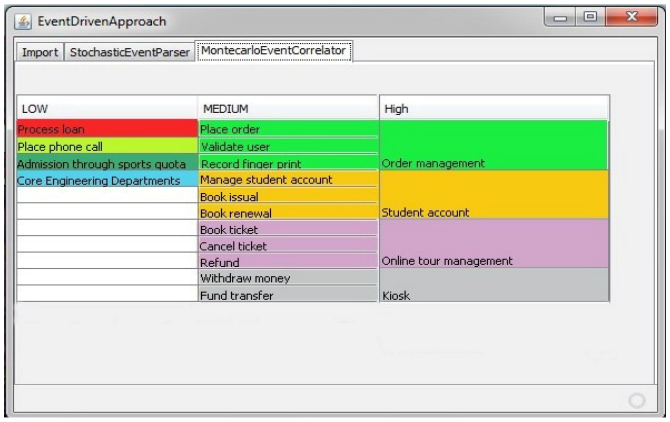

Figure: 8 Monte Carlo Event Correlator

Finally through Human Intervention the Correlated Events are optimized with a Single Event name. Those Optimized Events are shown in High prioritized column. Thus initially the System took 15 Events as input, among which Stochastic Event Parser classified 4 Events to be unique and 11 to be Correlated. Monte Carlo Event Correlator categorized the 11 correlated Events from parser to 4 Unique Events. We finally obtain optimized 8 unique functional Events. Thus the proposed Event Optimizer tool provides approximately 50% optimization.

CONCLUSION

Gathering Requirements is the most important task which needs to be planned ahead of time. There are several techniques to extract the Requirements from Clients. All those on an average require one fourth of the projects life time. After investing One fourth of a projects life time in Requirements gathering, requirements tend to change. On an average around 98% of projects till date have agile requirements. Investing huge time on a project whose requirements are agile is said to be a utter waste. This paper provides a solution to extract Requirements in short time and also helps in transitioning the Requirements from Requirements gathering phase to Design Phase. We also overcome the challenge of Event optimization by developing a tool that helps in optimizing all the recorded Events.

It is clear that the time of Gathering Requirements can be reduced by following the Event-Driven approach. The quality of the Use cases derived from the Event Table tends to increase by applying Event Optimizer technique.

The future work to this paper would be to optimize the Events from Monte Carlo Event Correlator without human Intervention. We avoided it since all the requirements gathering process available till date currently follow a manual approach. In future if there is a necessity to develop a feasible automated Requirements Gathering approach our model might serve the purpose.

REFERENCES

[1] Ralph R.Young, "Recommended Requirements Gathering Process",Northrop Gruman Information Technology, April 2002, pg.9-12.

[2] Young Ralph R., "Effective Requirements Practices", Boston Addison Weisely, 2001.

[3] Kotonya, Gerald, Ian Sommerville, Requirements Engineering: Process and Techniques", Chichester England John wiley & sons, 1998.

[4] Hooks, IVF, Kristin A.Farry, "Customer Centered Products:Creating Successful Products through smart Requirements Management" AMACOM, 2001.

[5] Dmitri Ilkaev, "The Project perfect white paper Collection-Handling Uncertainity in Project planning", pg.1-9.

[6] HubertF.Hofman, Franz Lehner, "Requirements Engineering as a Success Factor in Software Project, IEEE software, july 2001, pg.58-66.

[7] Roel wieringa, Neil Maiden, Nancy Maed Colette Rolland, " Requirements Engineering Paper Classification & Evaluation Criteria: a Proposal and a Discussion", Sponger, 2005,pg.102-107.

[8] Bush D.,"Modelling Support for Early Identification of Safety Requirements: A Preliminary Investigation, IEEE Conf. On Requirements Engineering,2005.

[9. Cristina Afors, Marilyn Zukerman Michaels, A quick accurate way to determine Customer needs" American Society for Quality, July 2001, pg.82-87.

[10] Sherry Ferell, Roger Heller, " Joint Application Development & Function point Analysis-The Perfect Match", A publication for Information Technologies Professionals, 2008, pg.1-5.

[11] Mohammad I.Muhairat, Rafa E.Al-qutaish, "An approach to derive Use cases from an Event Table", Internalnational Conference Software, parallel and distributed System, Feburary 2009, pg.33-38

[12] Ellen Goltesdiener, "Use Cases:Best Practices" , A Technical Discussion on Use Cases Best Practices by IBM, 2003, pg.3-18.

[13] Alistair Cockburn, "Writing Effective Use Cases" Addison-Wesley Longman, 2000.

[14] Gunnar Overguaard, Karen Palmkvist, Ivar Jacobson, "Use Cases: Patterns and Blueprints", Addison Weisely Professional, 2004.

[15] Grady Booch, James Rambaugh, Ivar Jacobson, "The Unified Modelling Language User guide", Addison Weisely Professional, 1998.

[16] jim Arlow, Ila Neustadt, "UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design", Addison Weisely Professional, 2005.

[17] Karl E. Wiegers, Sandra Mckinsery, "The proven way to accelerate Development" Serene Publications, pg.3-13.

[18] Micheal C.Fu, "Optimization for Simulation Theory Vs Practice", INFORMS Journal on Computing, 2002, pg.192-215.

[19] Andreas Muller, "Event Correlation Engine" ETH:Eidgenossicha Technische Hochushule Zurich, 2009, pg.6-57.

[20] Risto Vaarandi, Micheal R.Grimaila, "Security Event Processing with simple Event Correlator", ISSA Jounal, 2012, pg.30-37