Project Risk Management Using Event Calculus

Andreas Gregoriades, Vicky Papadopoulou Lesta, Petros Petrides
Dept. of Computer Science and Engineering,
European University Cyprus.

6, Diogenous Str., Engomi, P.O. Box: 22006, 1516
Nicosia-Cyprus
{a.gregoriades, v.papadopoulou}@euc.ac.cy,
petros1@eclatent.com

Abstract - **Risks are unavoidable in systems engineering projects due to emerging changes throughout the project's lifecycle. Changes in activities and peoples roles are usually not free from conflicts, which in some cases if not dealt adequately, increase the project failure risks and could bring the whole project to a standstill. Herein, we present a method that examines the impact of change to project's duration which constitute one of the most critical risks in project management. The method proposed utilizes two main criteria namely, the degree of dependency among activities and actors, and the temporal costs associated with the change. The proposed methodology is realised in Event Calculus (EC) and elaborated with an example.**

Keywords: Event Calculus, Project Risk Managment, Conflict Impact Analysis.

I. INTRODUCTION

Despite sophisticated tool support for project management, projects still continue to suffer by budget overruns and a failure to meet user requirements. Various approaches have been proposed as solutions. An example is the capability maturity model integration (CMMI) [6] that defines layers of control to help ensure quality and efficient procedures that yield an improved project development process. In particular, CMMI's change control, limits project's scope creep, while, the management review, enables intermediate quality checks on the system. These have shown to be effective in improving project performance. One indicator of project quality is process flexibility, that allows rapid and cost-effective modifications of new needs [7]. However, achieving a high level of project flexibility has often been slow, inflexible, and time-consuming, even with the aid of sophisticated tools and methodologies, which oppose the goal of efficient production. Therefore, it is necessary to examine higher levels of project development flexibility by evaluating risks while still meeting budgetary and temporal restrictions. These include, role management and requirements change during a typical system development life cycle (SDLC). Requirements change analysis is translated into change impact analysis that investigates the effect of change to dependent requirements, activities and people involved. However, in most of the times changes to requirements or roles during the SDLC are not free from conflicts. If these are not resolved they could lead to inconsistencies in the process that could result in product failure. Conflict analysis falls under the change management process that is composed of three conceptual phases, namely: dependencies specification, traceability methods, conflict analysis and resolution. The method proposed herein, demonstrates the application of EC to identify conflicts in project given specifications of project's schedule, tasks and modifications. Resolving conflicts is of critical importance since they can bring a project to a standstill. The proposed method can monitor the task completion of each team member, and check if the project schedule falls within agreed time limits. Changes to requirements are also considered to assess the extra time introduced.

A. Problem Definition

Project risk management encompasses knowledge, techniques, and tools necessary to manage the risks that could emerge during project's life cycle. It is a methodological approach to minimizing uncertainty that could jeopardize achieving agreed output upon specified time frame with defined resources. An important phase during project risk management is the definition of activities. These may be related to each other via dependencies, i.e., action A_i must be implemented before A_j is completed. Given these interdependencies, conflicts during the introduction of change creates emergent risks. The potential of conflicts in systems engineering projects is usually high due to the involvement of individuals from different backgrounds, specializations and locations, working together to achieve complex tasks. The objective of risk management is to provide the system designers/managers with early detection of risky situations as these are manifested by conflicts. Output from this process aids system designers in adopting appropriate countermeasures to resolve or at least mitigate emergent risks.

B. Contribution

During the development of a project, it is critical that some tasks are completed at specific times. However, actions or inactions of involved members may cause critical delays that could lead to failure of accomplishing critical tasks that lay on the critical path. However, during the development of a project, changes in the initial team structure or project specification could cause extensive overheads to the project's schedule. These changes could create risks of the following categories: critical requirements, people and estimation risks.

Herein, we propose a Project Risk Management tool that quantifies the effect of project changes and in essence provides valuable feedback regarding the project's estimation risk. During project execution, actions of each member are reported in the tool that detects possible conflicts due to changes in team structure. The tool resolves conflicts by suggesting appropriate modifications in the project implementation plan to improve the likelihood for guarantying success. The tool is based on EC. In particular, using appropriate event specifications and axioms, the tool identifies conflicts upon changes to project properties. Information that is used during the analysis includes: events relating to functional project requirements, temporal specification of project characteristics i.e. start, end date of the project, roles, capabilities and actions of project members and modifications to project specifications. Inference rules are used to identify conflicts caused by changes or actions

that do not adhere to the project plan. Conflicts resolutions are provided when feasible.

C. Related Work

Work by Giorgini et al [5] describes a conflict detection method, composed of three phases namely: modeling, extensional description and verification. During the modeling phase the designer draws the extensional requirements models where every node and edge corresponds to a fact in the formal framework. Then, the reasoning system completes the extensional description of the system using rules and verifies its consistency using constraints. If any inconsistency is detected by the reasoning system, the designer revises the requirements model to avoid or at least mitigate detected conflicts and repeats the formal analysis step. This method uses the Datalog solver which is based on Boolean algebra to detect conflicts. In contrast to this approach the methods described herein is not indented to work solidly on requirements analysis, but also during project implementation where changes are more expensive to be realized and alterations to project schedule more important. EC was also utilized in [5] for policy and specification and analysis. The proposed method transforms policy and system behavior into formal notation which is based on EC. Work by [8] and [4] also use EC as a specialised first-order logic for formalising policy specification and accordingly identify conflicts and propose resolutions in network and systems management. Chomicki [9], similarly use constraints which are policies that prevent a specified action from being performed in a given situation.

II. BACKGROUND

A. Event Calculus

EC allows specification of system behaviour using notations, such as, state charts, which can then be automatically translated into a logic program representation. EC supports deductive, inductive and abductive reasoning. Abductive reasoning proof for EC can be used to detect the existence of potential conflicts in partial specifications and generate explanations for the conditions under which such conflicts may arise [4]. EC is a "logical language" from which actions and their effects can be represented using predicates. Therefore, in system engineering projects, actions that took place at particular time using certain resources to produce specific output, can be defined, explained and proved using EC. Consider the following example: Maria was hired as an account in a company on May 11 2001, Maria was promoted to head accountant on May 11 2002, Maria left her position on 23 April 2004. Based on these statements we can make the following assumptions:

A) Maria was an accountant after May 11 2001.
B) Maria ended his accountancy post on May 11 2002,
C) Maria was a chief accountant from May 11 2002 until 23 April 2004.

If the events however were mentioned in the following manner: "Maria is hired as an accountant in a company on May 11 2001, Maria left the company as a chief accountant on 23 April 2004" then the assumptions cannot hold; There is a timing gap between the ranking of the person, therefore the promotion cannot be assumed, as other events could have happened such as maternity leave. Solution to this problem is divided into two phases: the theorem problem solving, which uses theorems that have been proven in order to give the solution and the problem specification and execution part, which uses logic to infere solutions. In logic programming the implications are treated as a base for goal reduction procedures.

B. Logic Programming & Prolog

Logic programming uses mathematical logic in computer programming. Prolog is a logic-based programming language developed by Alain Colmerauer, that use the following constructs: an *atom* that describes a general-purpose meaning, a number which can be floating or integer, a variable denoted by a string consisting of letters, finally, compound terms composed of "factor" which is an atom and the arguments. Prolog works by making *deductions* and *derivations* from facts and rules stored in a *database*. The essence of Prolog programming is writing crisp, compact rules. The deductions and derivations, instigated by user-entered queries, are products of Prolog's built-in inference mechanism called *backtracking* which is part of abductive reasoning.

III. METHODOLOGY

The Project Risk management tool is built using EC. In particular, using appropriate events, rules are utilized in order to monitor projects parameters, events related to a particular project implementation, such as the beginning date of the project, actions of members and changes to project specifications. Inference rules are used to backtrack the queries to identify conflicts caused by changes or actions that do not satisfy the project planning. Conflict resolutions strategies are presented when feasible.

A. Describing Events and Relationships with EC

Consider the following scenario: Two teams are working on systems engineering project involving the development of a car simulation project, which includes both hardware and software implementation. The project requires parallel development of both software and hardware. This procedure has the advantage of allowing various parts of the project to be progressed simultaneously. Testing can be done after completion of each phase. Team one consist of Martha, Tim, and Thomas. Martha has similar expertise acquired from similar training with Tim and can replace him if necessary. Tim is responsible for the gas system, Martha for the pipeline system and Thomas for the steering system. Give these descriptions, the following events hold:

- E1 is an event at which Tim is trained for the gas system. E1 has time 10 March 2004.

- E2 is an event at which Martha is trained for the pipeline system. E2 has time 5 March 2004.

Events are recorded in a database. Note that updates of the database are additive; events are added in the database and no deletions are allowed. In order to delete an event that expresses a relationship, a new statement needs to be set that ends the relationship. Chronological events are treated without any particular order, hence, older events can be added after adding

recent events. The chronological order of the events can be expressed using the <, >, = symbols. For instance, if an event E1 occurred before an event E2 then it can be described as E1<E2. Events can also have duration, a starting point and an ending point.

1) Representation

Given the systems engineering example, an event concerning the ranking of a person in the project, is expressed in EC as follows:

X has rank Y for a period E, provided that X has done something at a period that is equal to E.

This means that after an event happened, variable X has rank Y for a chronological period E, which is actually equal to the time following E. In this example let's assume that Tim started working on the project on the 21st of April 2004. Therefore E1 is an event with timing: 21 of April 2004, at which X has rank Y (Tim is working on project) for the period after E, that is after the 21st of April. The events E1, E2 are in the past and their time period will finish when the person is assigned on a different rank. Events are expressed in EC using the following form: (*i*) Time (Event, Time), to express that event Event happened at time Time. (*ii*) Trained(Martha, pipeline system), to express the person Person was trained for activity A and (*iii*) Rank (Person, Time, trained A), to express that the person Person at time Time has been trained for activity A. So, for example, events E1, E2 can be expressed as follows: Trained(Tim, gas system). The *Hold* predicate is used in order to express the time periods of the relationships. The *Holds(p)* predicate states that a relationship associated with P holds for a time period p. Likewise the statement,

Rank(Tim, working on gas before (E2))

can now be written as

Holds(before (E2 rank (Tim working on gas))

which means that "Tim holds the rank of working on gas for a time period which is before E2, or in other words which last before E2 starts."

Additionally, the following predicates state:

- Initiates: represent the start of an event.

- Terminates: represent the termination of an event.

- Broken: represent identical or incompatible relationships.

Therefore, Holds(before(e u)) IF Terminates(e u), implies that "An event E holds before time period u, if the event E is terminated at time period u".

IV. Example Application of EC in a Systems Engineering project

Application of the approach is demonstrated with an example from the systems engineering domain. In particular, we elaborate on the approach through an example drawn from the development of a vehicle subsystem. The process commences with the population of the database with the temporal project activities relating to the functional system

requirements and their dependencies. Next, a second database of *actors specializations* is created. Finally, the "*event planner*" database is specified that holds the tasks of each actors as scheduled by the project plan. New events can be added in the database during project execution and while monitoring the project's progress. During project execution changes can be instantiated and subsequently evaluated using queries relating to activities or roles in the project schedule. The tool provides the user with the impact and feasibility of each change.

A. Project Specification

Consider the following model of a project specification depicting *activities* A0, A1, A'0, B1 and their dependencies. The diagram is read as follows: Activity A2 should be implemented before A1 and activity A1 should be implemented before A0, and B1 should be implemented before A'0. The dashed line means that Activity A1 can be replaced by activity B1.
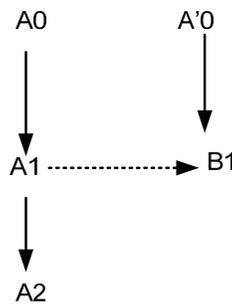
B. Actors

Participants in the project, called *actors* are involved in the development of the project by undertaking activities, provided they are trained prior to engaging with the activity. Therefore, the training of actors P1 and P2 could be set as follows:

- Actor P1 is trained for the implementation of A0, A1, A'0, B1 and A2.

- Actor P2 is trained for the implementation of A0, A1, A'0 and B1.

C. Events

The following events E0-E3, that took place during the development of the project are recorded in the database:

- E0 is the event that defines the project start date. Date is 10 Feb 2010.

- E1 is the event stating that actor P1 completes activity A0. Date is 11 Feb 2010.

- E2 is the event stating that actor P1 completes activity A1. Date is 16 Feb 2010.

- E3 is the event stating that actor P1 leaves from project for 4 days. Date is Feb 21 2010.

Training of each actor is recorded in the database in the following manner:
- Rank(P1,(before E0),trained A0)) ^ Rank(P1,(before E0),trained A1)) ^

- Rank(P1,(before E0),trained A'0)) ^ Rank(P1,(before E0),trained B1)) ^ Rank(P1,(before E0),trained A2)).

which is translated to:
"P1 will be trained for activities, A0,A'0,A1,B1,A2 at time period before E0".
Also, we record:
- Rank(P2,(before E0),trained A0)) ^ Rank(P2,(before E0),trained A1)) ^

- Rank(P2,(before E0),trained A'0)) ^ Rank(P2,(before E0),trained B1)) .

which is translated to:
"P2 will be trained for activities, A0,A'0,A1,B1 at time period before E0".

D. Conflicts Identifications and Resolution

After recording all events related to the project, we can make queries concerning the temporal feasibility of the project. In particular, we can make queries of the following form:

- rank_of (P,Y,T): to get the rank of actor P at time T. The system will report the actions Y performed at that date.

Moreover, to find the rank of actor P, on the 16th of Feb 2011 we make the following query: "rank_of((P1,Y,(16 Feb 2010))".The system will search the database for the rank of actor P1 and will return the actions Y based on which the rank of P1 on the specified date is satisfied. Hence the output would be: "implements activity A0". When a project manager is interested to change an activity in the project plan, the following query could be used to get the dependencies of an action A:

- Activities_of(A, Y)

Similarly, in order to inquire about the training of actor P, on activity Y, for time T we can use the following statement:

- Rank(P,T,trained Y):.

Therefore, to check whether a change in activity A'0 and B1 is free from conflicts, the following queries are made:

- Rank(P1,(before E0),trained A'0)),

- Rank(P1,(before E0),trained B1))

In this case the system responds with a "TRUE" clause which stipulates that we can safely assume the change of P1 without any conflict.

E. Examples of Queries

1) Example1

At time period E0, if there is a request for a change in activities A1 to B1, then this will result in no conflict: This however is the best case scenario since the project has just began and none of the activities have been implemented. In this case , the following predicate holds:
Holds (before E0 (P1 implements A0)) FALSE

Which means that the predicate "actor P1 has implemented activity A0 before event E0" does not hold.

2) Example 2

Similarly, if there is request for a change to project activities, on the 21st of Feb, E4, then this will result in a conflict, since no actor would be available to take over the change. Hence,

- Initiates (after E3 and before E4(P1 implements A0)) FALSE

- Initiates (after E3 and before E4(P2 implements A0)) FALSE

To resolve this conflict: Actor P2 can replace actor P1 and implement the required parts. A conflict however, may occur, when the activity A2 needs to be implemented at a time prior to actor's P1 completion of training :Rank(P2,(before E0),trained A2)) FALSE. This means that "Actor P2 is NOT trained for activity A2".

V. CONCLUSIONS AND FUTURE WORK

The work presented herein addresses an important problem in project risk management, namely conflicts analysis and resolution. The method described elaborates on the pressing need for timely delivery of projects within agreed time limits. The complexity and uncertainty of modern systems however makes this a challenging task. The statistics of project failures alone highlights the complexity of the problem. The method described provides the mechanism for an effective mitigation of project overrun risk that emerge with changes in project requirements, activities and roles using EC. This enables automated inference of project's state given instantiations of events that define changes to schedule, requirements or structure of the project plan Preliminary results from this work are encouraging. However, the full extent of the method will be realized with its thorough validation. On the same vein, project cost is also another burden to project managers that effectively could bring projects to standstill if not managed adequately.

VI. REFERENCES

[1] R. Kowalski., "Legislation as Logic Programs", Springer-Verlag, 1992.
[2] D. Kowalski, Kuehner, "Linear Resolution with Selection Function, Springer-Verlag, 1983.
[3] R. Kowalski, "Predicate Logic as Programming Language", EEE Computer Society Press, 1986.
[4] A. Bandara,E. Lupu, E. Lupu and A. Russo, "Using Event Calculus to Formalise Policy Specification and Analysis", Policies for Distributed Systems and Networks, 2003.
[5] P. Giorgini, F. Massacci, J. Mylopoulos, N. Zannone, "Detecting Conflicts of Interest", Proceedings of the 14th IEEE International Requirements Engineering Conference, 2006.
[6] D. Ahern, A. Clouse and R. Turner, CMMI distilled, Addison-Wesley, 2003.
[7] B. Hugher, Cottetell M. "Software Project Management", McGrahill, 2011.
[8] M.Charalambides, P. Flegkas, G. Pavlou, A. Bandara, E. Lupu, A. Russo, N. Dulay, M. Sloman, J. Rubio-Loyola. "Policy Conflict Analysis for Quality of Service Management", 6th IEEE International Workshop on Policies for Distributed Systems and Networks, pp. 99-108, 2005.
[9] Jan Chomicki, Jorge Lobo, Shamim A. Naqvi, "A Logic Programming Approach to Conflict Resolution in Policy Management", Proceedings of the 17th International Conference Principles of Knowledge Representation and Reasoning, pp 121-132, 2000.