A Goal Driven Framework for Software Project Data Analytics

George Chatzikonstantinou[1], Kostas Kontogiannis[1], and Ioanna-Maria Attarian[2]

[1] National Technical University of Athens, Greece
gchatzik@cslab.ece.ntua.gr, kkontog@softlab.ntua.gr
[2] IBM Toronto Laboratory, Canada
ioannaat@ca.ibm.com

Abstract. The life cycle activities of industrial software systems are often complex, and encompass a variety of tasks. Such tasks are supported by integrated environments (IDEs) that allow for project data to be collected and analyzed. To date, most such analytics techniques are based on quantitative models to assess project features such as effort, cost and quality. In this paper, we propose a project data analytics framework where first, analytics objectives are represented as goal models with conditional contributions; second, goal models are transformed to rules that yield a Markov Logic Network (MLN) and third, goal models are assessed by an MLN probabilistic reasoner. This approach has been applied with promising results to a sizeable collection of software project data obtained by ISBSG repository, and can yield results even with incomplete or partial data.

Keywords: Software engineering, software analytics, conditional contributions, probabilistic reasoning, multi-view goal models.

1 Introduction

The life cycle of large industrial software systems encompasses a number of diverse and complex tasks. The software engineering community has responded to this challenge by proposing environments that utilize software repositories to store a large collection of software artifacts and project related information. This information can be mined to provide the springboard for what is referred to as software development analytics, an area that has started receiving significant attention over the past year [1],[2].

However, current software mining techniques and tools are mostly used for knowledge discovery, and for the identification of relationships across repository artifacts [3], failing to take into account the project's contextual information that leads to different views of analysis and mining objectives. In this paper, we propose an approach where software project data analytics is taking the form of specifying, and consequently verifying or denying, specific hypotheses or goals regarding the risks related to cost, effort, and quality of the software system

C. Salinesi, M.C. Norrie, and O. Pastor (Eds.): CAiSE 2013, LNCS 7908, pp. 546–561, 2013.

being built or maintained. We consider the approach as being qualitative, even though it utilizes quantitative data for its training purposes, because it is based on goal models, instead of a numerical formula to compute its results. Existing cost, effort, and quality prediction models that are based mostly on numerical formulas fail to take into account experience captured from past similar projects, or to formally represent the view an organization and its stakeholders have on how risk should be defined and evaluated within the particular software development context. The intended use of the proposed framework is first, to allow stakeholders to define their own views on how risk related to cost, effort, and quality is to be modeled and evaluated; second, allow for past cases to be used for training the risk assessment models allowing thus for customization within an organization and third, allow for risk assessment to commence even when not all input values are known.

More specifically, we present a software project data analytics framework that is based first, on goal models for denoting analysis objectives and second, on Markov Logic that allows for reasoning even in the presence of incomplete or partial information. Furthermore, *a*) we enrich goal models with the concept of conditional contributions among goals initially introduced in [4], *b*) we illustrate the association of conditional contributions with agent roles and commitments which are originally presented in [5], and *c*) we provide the necessary transformation rules that allow for the generation of logic formulas and the corresponding Markov Logic Networks (MLNs) from such goal models. The produced MLNs can then be trained by past project data so that, probabilistic reasoning weights can be computed for the logic formulas. Consequently, current project repository data can be utilized, in order to produce analysis results as to whether these project objectives can be satisfied within a certain level of confidence.

This paper is organized as follows. Section 2 provides a research baseline by summarizing key concepts in the areas of Goal Models and Markov Logic Networks, and also describes the details of our approach. Then, section 3 discusses the concept of conditional contributions as it is used in this paper, the transformation of Goal Models to Markov Logic rules, and the training process. Section 4 presents a case study that uses goal models pertaining to quality, effort and cost. Finally, section 5 presents related work and Section 6 concludes the paper and provides pointers for future research.

2 Related Research Fundamentals

2.1 Goals and Commitments

AND/OR Goal Trees. The Goal Tree model is based on the concept of top-down decomposition of goals into subgoals and has been successfully used for specifying functional and non-functional requirements of software systems [5].

More specifically, a goal can be divided into sub-goals which are represented as its children. Borrowing the notation used in [5], we denote an AND-decomposition or an OR-decomposition of goal α to a set G of sub-goals as:

$$G \xrightarrow{AND} \alpha \qquad \text{and} \qquad G \xrightarrow{OR} \alpha \tag{1}$$

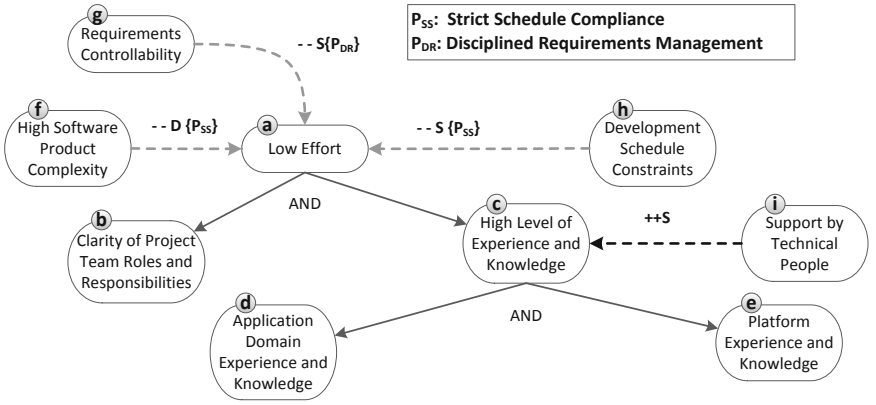

Fig. 1. An example goal model for "Low Effort" with conditional contributions

Except from AND/OR-decompositions, two goals can by connected by a contribution arc. In a more descriptive manner, a goal may potentially contribute to other goals by four different contribution arcs [6], namely, $++S$, $--S$, $++D$, and $--D$, which according to [5] can be expressed in mathematical logic as follows:

$$\begin{aligned} ++S(g,g') &\Rightarrow g \rightarrow g' & ++D(g,g') &\Rightarrow \neg g \rightarrow \neg g' \\ --S(g,g') &\Rightarrow g \rightarrow \neg g' & --D(g,g') &\Rightarrow \neg g \rightarrow g' \end{aligned} \tag{2}$$

It is important to note that for this paper we allow for *conditional contributions*, i.e. contributions that can be used only when a specific condition holds. Conditional contributions were initially introduced in [4] as part of a general contextual requirements modeling framework. In addition to that, contributions can be considered with a degree of probability which is called *weight*, and is computed via an appropriate training process. We believe that the existence of weights for contributions, and hence for the corresponding logical formulas, captures best the dichotomy between contribution of one goal to another and the strict notion of an implication in a FOL formula.

An example goal model [1] that contains only one root node, namely "Low Effort", and which is further AND-decomposed to sub-goals, is illustrated in Fig. 1. As it can be seen, in addition to the $++S$ contribution from node-i ("Support by Technical People") to node-c ("High Level of Experience and Knowledge") which applies under any case, there are also three conditional contributions. Two of them, $--D(f,a)$ and $--S(h,a)$, apply only when the *policy* chosen is one that demands strict schedule compliance, while the third one, $--S(g,a)$, applies only when a disciplined requirements management policy is adopted.

Agents, Capabilities and Commitments. Goal trees can also be used in multi-agent environments, where each agent owns a set of goal trees and aims

[1] This example is based on the analysis presented in [7].

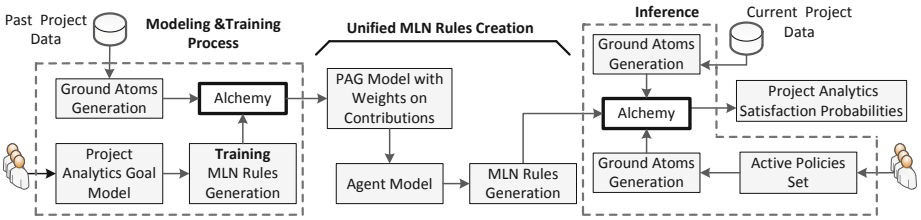

Fig. 2. The proposed framework for software project data analytics

at satisfying the root goals. An extension of this formulation is proposed by Chopra et al. in [5], where the authors are aiming to model the communication protocol between agents with different goals. The novelty of this approach is the introduction of three new concepts, namely *Capability*, *Commitment* and *Role*.

In a more descriptive manner, each agent has a set of *Capabilities*. Those are the goals that the agent can achieve without the need to interact with other agents. There are some goals however, the satisfaction of which depends on other agents. This kind of dependency is described by the notion of Commitment. More specifically, a Commitment is a tuple:

$$Com(Debtor,\ Creditor,\ Antecedent,\ Consequent)$$

which means that the *Debtor* is committed to the *Creditor* for the *Consequent* if the *Antecedent* holds. The Debtor and the Creditor in the previous tuple are Roles that an agent can adopt, while the Consequent and the Antecedent, in the cases used hereinafter, are goals of the model.

2.2 Markov Logic Networks

Richardson and Domingos [8] have introduced MLNs as a way to combine the benefits of both first-order logic (FOL) and probabilistic graphical models in a single representation. More specifically, an MLN constitutes a knowledge base (KB) of predicates and ground atoms represented as nodes in a graph. By assigning truth values to each possible ground atom, possible worlds can be constructed in which the KB may be true with a degree of probability. In MLNs, a world may hold with a non-zero probability, even if some of its formulas are violated. The degree of probability for the world to hold, depends inversely on the number of formulas violated and also, on how strong the constraints introduced by the violated formulas, are. The latter is signified by a real number related to each formula in the KB, which represents the weight assigned to this formula.

For this paper, we have used the Alchemy tool as a statistical relational learning and probabilistic logic inference engine [9]. By providing a set of MLN rules along with the set of grounded atoms, Alchemy tool constructs the appropriate Markov network which can then be used either for training or for inference.

2.3 Process Outline

The outline of the proposed framework process is depicted in Fig. 2. Initially, the analysis objectives are defined and represented as a goal model with conditional contributions which is called Project Analytics Goal (PAG) model. This is constructed by using an appropriate visual notation, the semantics of which are described in section 3.1. The presence of conditional contributions allows for the existence of multiple variations of the same model, with each variation capturing a particular model viewpoint. In a similar manner as in [10], conditional contributions provide the flexibility of manually turning on or off specific contributions according to the need of the analysis. For example, for the PAG model illustrated in Fig. 1, the two conditional contributions $--D(f, a)$ and $--S(h, a)$ may or may not be used in the analysis, and that depends on whether policy P_{SS} is true or not. Consequently, the given PAG model is transformed to a set of rules that form a knowledge base for a Markov Logic Network (MLN). The resulting MLN can then be trained by past project data so that, probabilistic reasoning weights can be computed for rules that correspond to contributions. It is important to note, that during the training process we assume that all contributions, whether they are conditional or not, are active. This assumption is based on the fact that the degree of probability some objective contributes to an other objective is specific, and it does not depend on the viewpoint of the model.

The training process described so far, produces an updated PAG model in which weights, i.e. degrees of probabilities, have been assigned to each contribution. The role those weights play in the analysis can be pointed out through the following example. Consider the case of node-c in Fig. 1 which is the target of a $++S$ contribution from node-i. The existence of this contribution implies that in case node-i is true, node-c can be satisfied with some degree of probability, even if its child nodes, namely node-d and node-e are false. As soon as the training process completes, and the updated PAG model has been generated, we create the set of MLN rules that are going to be used by the inference engine. However, the presence of conditional contributions implies that we have to generate a different set of rules for each possible combination of policies, i.e. for each potential viewpoint. To avoid this, and in order to produce a unified set of rules, we consider an *Agent Model*, which uses the notion of commitments as described in [11]. The equivalence between conditional contributions and commitments is described in detail in section 3.2.

Finally, current project repository data, and the required viewpoint in terms of active policies, are fed as input to the framework, in order to produce analysis results as to whether the project objectives can be satisfied within a certain level of confidence.

3 Modeling Project Analytics Goals

3.1 Policies and Conditional Contributions

Policies provide the mechanism to restrict the contributions that may be active at any given point, reflecting thus the different viewpoints agents have on a goal

Table 1. Equivalence between conditional contributions of a PAG model and commitments in an Agent Model. C_P are the PolicyAgent's capabilities.

PAG Model	**Agent Model**	**Interpretation** ***(if p holds)***
$w : n_s \xrightarrow[\{p\}]{++S} n_t$	$Com(R_P,p,n_s,n'_t)$, $n'_t \in C_P$ $w : n'_t \xrightarrow{++S} n_t$	n_s contributes *positively* to n_t **with probability** w
$w : n_s \xrightarrow[\{p\}]{--S} n_t$	$Com(R_P,p,n_s,n'_t)$, $n'_t \in C_P$ $w : n'_t \xrightarrow{--S} n_t$	n_s contributes *negatively* to n_t **with probability** w
$w : n_s \xrightarrow[\{p\}]{--D} n_t$	$Com(R_P,p,n'_s,n'_t)$, $n'_t \in C_P$ $n_s \xrightarrow{--D} n'_s$, $w : n'_t \xrightarrow{++S} n_t$	n_s denial implies the *achievement* of n_t **with probability** w
$w : n_s \xrightarrow[\{p\}]{++D} n_t$	$Com(R_P,p,n'_s,n'_t)$, $n'_t \in C_P$ $n_s \xrightarrow{--D} n'_s$, $w : n'_t \xrightarrow{--S} n_t$	n_s denial implies the *denial* of n_t **with probability** w

model given the project's context or characteristics. Hence, policies allow for the existence of multiple views of a Project Analytics Goal (PAG) model, where each combination of policies yields a different viewpoint that contains a different set of active contributions. The use of conditional contributions provides a modeling abstraction to the Goal Models theory. More specifically, the conditional contributions allow for a user to define the context in which contributions hold. Once the context (i.e. the conditions) have been defined by the software engineer, then only the contributions that their values evaluate to true are considered, resulting thus to a fully unconditional Goal Model which is a "specialization" of the original one containing the conditional contributions. The training process assigns probabilities to the contributions that are conformant to the semantics of Goal Models with contribution probabilities as these are defined in [12]. What follows are the formal definitions of conditional contributions and PAG models.

Definition 1 *Let $T \in \{++S, --S, ++D, --D\}$ be the type of a contribution. Let also P_F be the set of all possible policies for a given PAG model, and $P_C = \{P_{c_1}, \cdots, P_{c_n}\} \subset P_F$. A* conditional contribution *from a to b, denoted as a $\xrightarrow[P_C]{T} b$, is a contribution of type T that applies only for policies in P_C.*

Definition 2 *A PAG model is a triplet of the form $\langle G, P_F, G_S \rangle$, where G is the set of decomposition and contribution statements, P_F is the set of all possible policies and G_S is the set of the conditional contributions of the model.*

It is important to note that for a contribution to be conditional, P_C has to be a proper subset of P_F, as otherwise it would be possible to define conditional contribution that would apply to every view of the model. Those conditional contributions, if they could be defined, would be degenerated into unconditional contributions which are denoted as $a \xrightarrow{T} b$ with T being the type of the contribution. For the PAG model illustrated in Fig 1, the corresponding P_F, G and G_S sets are defined as follows:

$$P_F = \{P_{SS}, P_{DR}\}\ ,\ G = \begin{cases} \{b,c\} \xrightarrow{AND} a \\ \{d,e\} \xrightarrow{AND} c \\ i \xrightarrow{++S} c \end{cases}\ ,\ G_S = \begin{cases} f \xrightarrow[P_{SS}]{--D} a\ ,\ g \xrightarrow[P_{DR}]{--S} a \\ h \xrightarrow[P_{SS}]{--S} a \end{cases}$$

When conditional contributions in G_S are activated these produce a *view* of the model. To accomplish this, a set of policies, we refer to as *active policies set*, must be specified. This set is denoted as P_A and is a subset of P_F. Given a conditional contribution in the form $a \xrightarrow[P_C]{T} b$, we say that this contribution is active during the analysis process only if $P_C \bigcap P_A \neq \emptyset$. For example, for the model in Fig. 1, if $P_A = \{P_{SS}\}$ the resulting view will contain all contributions except the one applied for policy P_{DR}, i.e. the $--S$ contribution from node-g to node-a, as $\{P_{DR}\} \bigcap P_A = \emptyset$.

3.2 From Conditional Contributions to Commitments

In this section, we discuss the association between *conditional contributions* and commitments, as the latter were originally defined in [11]. Assuming that the training process has been completed (see section 3.3), and that for each conditional contribution a weight, i.e. a degree of probability, has been calculated, the use of commitments aims at proving the abstraction means for creating a model that contains no conditional contributions and is still able to represent the multiple views of the initial PAG model in a single and unified rule base. Hence, our objective is not only to create a model that is consistent with the general goal model theory, but also to be able to use that model for the generation of a set of MLN rules that contains all rules and predicates to be used by any collection of policies (i.e. viewpoints) selected by the user without the need to regenerate rules for each viewpoint. In this context, commitments provide an abstraction mechanism that is compatible with goal model theory and are used in our approach to generate rules that act as switches to inhibit or prohibit reasoning paths depending on the policies selected. Specifically, we introduce two agents, namely the *ProjectAgent* and the *PolicyAgent*. We assign the goal model to the *ProjectAgent* while the *PolicyAgent* is used in order to provide certain objectives to *ProjectAgent* through commitments. Additionally, given a PAG model $\langle G, P_F, G_S \rangle$ we create one role for each policy in P_F and we allow only *ProjectAgent* to adopt one or more of these roles. This, in combination with one extra role, called *Reasoner*, denoted as R_P and adopted by the *PolicyAgent*, allows the definition of the required commitments. For example for the model of Fig. 1, ProjectAgent can adopt the roles P_{SS} and P_{DR}. Each combination of these adopted roles will result in the activation/deactivation of the corresponding contributions, and this is done through commitments.

Table 1 summarises the association between conditional contributions and commitments, where the degree of probability for each contribution, as this has been calculated during the training process, is denoted by w. Because of space limitations we only discuss the transformation steps and the use of commitments

for one of the four contribution types, namely the $++D$ conditional contribution. Initially, we introduce two pseudo-objectives, namely n'_s and n'_t, with the latter being a capability of the PolicyAgent. Subsequently, we add two contributions, one of type $--D$ from n_s to n'_s, and an other one of type $--S$ from n'_t to n_t. While the former is a contribution that applies always, the latter has a degree of probability equal to that of the corresponding conditional contribution. Consequently, we consider the commitment $Com(R_P,p,n'_s,n'_t)$. By considering this commitment, is equivalent as of adding the rules $p \wedge n_s \rightarrow n'_s$, $n'_t \rightarrow n_t$, where the former acts as the "switch" to inhibit or prohibit the generation of n_t whether p holds or not. Therefore, commitments provide abstraction and modeling means for representing a unified knowledge base. One could omit the use of commitments and generate directly the intermediate switch rules but we believe that the use of commitments provides a better and more abstract way to capturing the intended policies that can be adopted by the ProjectAgent.

Finally with respect to the example above, we illustrate the equivalence between the semantics of the $++D$ conditional contribution, and the rules that are produced by the use of the commitment. In particular we show that when policy p holds, the denial of n_s implies the denial of n_t with probability w. Actually, if n_s is false, then because of the $--D$ contribution n'_s becomes true. This means that in case ProjectAgent adopts role p and because of the commitment previously specified, PolicyAgent will provide ProjectAgent with n'_t which is one of PolicyAgent's capabilities. Finally, the truth of n'_t implies with probability w that n_t is false because of the $--S$ contribution introduced earlier. Hence, if n_s is false and ProjectAgent adopts the role p, objective n_t become also false with probability w. Thus, by substituting the initial $--D$ *conditional* contribution with an appropriate set of Agent Model constructs, we end up with a model that does not contain this conditional contribution but still has the same behavior as if it was part of the model.

3.3 Rule Generation for Training and Inference

As it is illustrated in Fig. 2, the generation of the MLN rules is required both for training and for inference. For the former, we generate the rules from the PAG model, for which we assume that all policies hold, i.e. all conditional contributions are active. Hence, we must extract a set of first order rules from a goal model that only contains contributions (non-conditional ones), and AND/OR-decompositions. This can be easily done by using only one first-order logic predicate, namely *Satisfied*(a), which means that objective a is satisfied. More specifically, the contribution of the form $--S(h,a)$ of Fig. 1 will be translated into the formula :

$$\text{Satisfied}(h) \rightarrow \neg\text{Satisfied}(a) \tag{3}$$

while corresponding formulas are used for the remaining contribution types. Furthermore, AND/OR-decompositions are translated as logical conjunctions/disjunctions. The AND-decomposition $\{d,e\} \xrightarrow{AND} c$ of Fig. 1 for example will be translated as :

$$\text{Satisfied}(d) \wedge \text{Satisfied}(e) \rightarrow \text{Satisfied}(c)$$

While the rules that correspond to AND/OR-decompositions are assumed to be hard, weights must be calculated for those that correspond to contributions, which is done through the training process.

In contrast to the previous case, the MLN rules required for inference are extracted from the produced Agent Model, which in addition to contributions and decompositions includes commitments and roles. For the former we define the predicate *Commit*(R_P,p,a,b), which means that *Com*(R_P,p,a,b) exists in the Agent model, while for the latter the *UsesPolicy*(p) predicate is used to denote that PolicyAgent has adopted role p, i.e $p \in P_A$. Finally, the following rule, resolving satisfaction of an obligation through a commitment must be added to the produced set of rules :

$$\text{Commit}(R_P, p, a, b) \wedge \text{UsesPolicy}(p) \wedge \text{Satisfied}(a) \rightarrow \text{Satisfied}(b)$$

4 Evaluation

4.1 Software Project Analytics Goal Models

As a proof of concept we have compiled a PAG model pertaining to Product Quality, Project Cost and Project Effort. Even though the compilation of this model reflects an agent's views and therefore it is subjective, we have attempted, when drafting our own model, to take into account assertions from the related literature as well as from existing standards. More specifically, with respect to product quality we referred to the ISO 9126 standard. With respect to cost and effort we have considered features (but not the actual metrics) from cost and effort estimation tools such as COCOMOII, PRICE-S and, CHECKPOINT. However, these goal models are not the primary focus and contribution of this paper as they are indicative and introduced for experimentation purposes. In this respect, a user may define his or her own models using the modeling principles introduced in this paper. A part of the PAG model utilized for the case study is depicted in Fig. 3. The conditional contributions in these models pertain to the policies: *Strict Adherence To Process Model Analysts Policy* (depicted as PM in Fig. 3), *Strict Adherence to Coding Standards Engineering Policy* (depicted as CS), *Strict Organizational Structure Management Policy* (depicted as OS), *Use of High Level Language Analysts Policy* (depicted as HL), and *Extra Attention to be Paid For Large or Legacy Systems Management Policy* (depicted as LS).

4.2 Case Study

In this section, we present experimental results obtained by applying the proposed approach to 280 different projects selected from the ISBSG [13] project data repository using three criteria. The first criterion is that we have opted

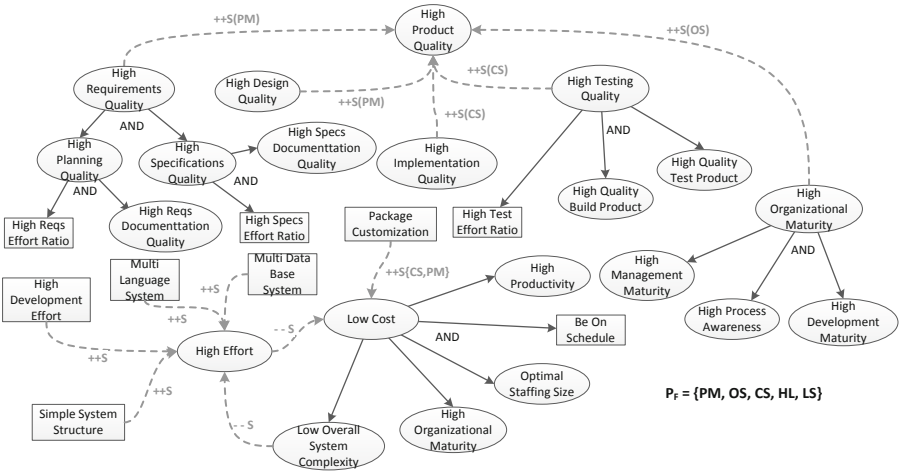

Fig. 3. Part of the PAG model used in the case study along with the set P_F of possible policies. Leaf nodes are depicted as rectangles.

for A Class projects meaning that, according to the ISBSG quality rating classification, the reported data is sound with nothing being identified that might affect the integrity of the analysis. The second criterion is that we have opted for projects that have been classified as *Development Project* so that we have kept the analysis related to the same type of projects. The third criterion is that we have selected large projects that correspond to the top 25^{th} percentile of projects with the highest Unadjusted Function Count as measured by the IFPUG standard. Each project has 44 distinct features with numeric or yes/no values. From these values we have created predicates that populated the Knowledge Base as follows. Values that belong to the top 25^{th} percentile of values were classified as *High*, values that belong between the top 26^{th} - top 75^{th} percentile were classified as medium and values below the top 75^{th} percentile as low. For example, if for a project the *Normalized Work Effort* attribute has a value that belongs to the to 10^{th} percentile of all 280 projects considered, the predicate *Satisfied(High_Nomalized_Work_Effort)* is added to the KB. For the training we have selected 30 projects representing the projects that had more than 80% of all their fields completed with values. In this sense, we allowed for the most complete MLN training we could get for the given ISBSG data set. From the remaining 250 projects we excluded the ones that had more than 60% of their values unknown for each category resulting in a case study data set of 246, 221, and 246 for the effort, cost, and quality respectively.

Policy Variability. Table 2 illustrates the variation of these probabilities as a function of the number of different used policies. It is interesting to note that when *AttentionForLargeSystems* policy is assumed the *LowCost*, *HighEffort* and *HighQuality* goals are satisfied with probabilities 21.57%, 99.04%, and 49.57%

Table 2. Variation of Probabilities with the allocation of policies

Policy	Low Cost	High Effort	High Quality
AttenForLargeSystems	21.57 %	99.04 %	49.57 %
HighLevelProgLanguageUsed	99.00 %	77.69 %	50.76 %
StrictCodingStructure	19.13 %	98.99 %	87.00 %
FollowProcessModel	20.13 %	99.04 %	83.59 %
StrictOrgStructure	19.13 %	99.00 %	99.00 %

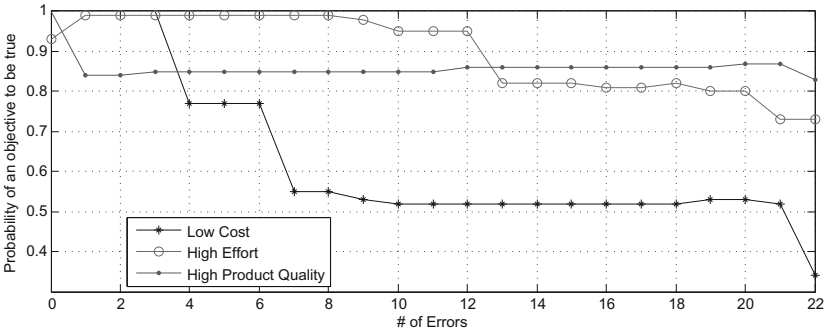

Fig. 4. Variation of probability while the number of erroneous features increased

respectively, but when *HighLevelProgLanguageUsed* is assumed as a policy the *HighQuality* goal probability increases, while the *LowCost* goal probability increases and *HighEffort* goal probability decreases reflecting the fact that the use of a high level language facilitates and eases development.

Stability. Experimental results indicate that the process is stable when reasoning commences with erroneous information. Fig. 4 illustrates the gradual decrease of probability result values as more and more features are excluded (negated). More specifically Fig. 4 depicts the effect of removing 0, 1, all the way up to half of features used for reasoning. Removing (negating) more than half of the features results into uncertain reasoning as not enough information is available to deduce an accurate result.

Correctness. To assess the correctness of the results we consider three specific numerical evaluation criteria (*Defect_Ratio*, *Effort_Ratio* and *Cost_Ratio*, that are indicative and fit to each goal type (product quality, effort, cost). These are consequently normalized to *Low*, *Medium* and *High* values, so that we can compare them against the obtained results by the goal models which also fall in

the *Low*, *Medium* and *High* scale according to their computed probability values. More specifically, for assessing product quality we have selected the

$$\text{Defect\_Ratio} = \frac{\text{Total Defects Delivered}}{\text{Functional Size}}$$

where *Functional Size* is measured in Unadjusted Function Points using the IFPUG standard. Normalized defect counts with respect to functional size (not code size) have been always associated with quality in the software engineering literature [14]. To evaluate whether this metric was a good validator we have run a Chi-square dependency test with four degrees of freedom (LowQuality, MediumQuality, HighQuality, and LowDefectRatio, MediumDefectRatio, HighDefect Ratio) and confidence level p = 0.10 for assessing whether there is any significance of DefectRatio in Quality. Our H_0 hypothesis that there is no significant relation between Defect_Ratio and Quality, was rejected with a Chi-square value of 8.75 corresponding to a probability of 0.93. The High, Medium, Low value classification for Defect_Ratio is based on the percentile ranking of obtained Defect_Ratio values (top 33%, top 66%, and low 33% respectively), while for the values of product Quality objective (as those are calculated using the framework), the High, Medium, and Low classification is based on $p > 0.75, 0.5 < p \leq 0.75, p \leq 0.5$ values respectively. Similarly, for assessing effort we have selected the

$$\text{Effort\_Ratio} = \frac{\text{Summary\_Effort}}{\text{Functional Size}}$$

that has given a four degrees of freedom Chi-Square value of 6.91 rejecting thus the null hypothesis with a corresponding probability of 86%. Finally for assessing cost we have selected the

$$\text{Cost\_Ratio} = \frac{\text{Project Elapsed Time}}{\text{Functional Size}} \cdot \text{Median Salary}$$

where median salary is estimated at 58,000 USD based on US national averages. As above, values in the top 33%, top 66%, and low 33% have been classified as High, Medium and, Low respectively. The *Cost_Ratio* metric has given a four degrees of freedom Chi-square value of 15.7 rejecting thus the H0 hypothesis that there is no significant relation between *Cost_Ratio* and the *Low_Cost* goal. Please note that for the evaluation of goals in the goal tree, we have not used any feature that is also used in the three validation criteria metrics presented above. The percentage of correct results as well as of false negatives and false positives for all three goals (i.e. High_Product_Quality, Low_Cost, High_Effort), obtained from the selected ISBSG projects excluding the projects used for the training set, are illustrated in Table 3. In a more descriptive manner, the framework predicts High_Product_Quality, Low_Cost, and High_Effort for a specific project when the corresponding probability calculated is greater that 0.75 and it is of medium confidence when the probability is between 0.5 and 0.75. Overall, the conclusions drawn from the projects analysed is that after training commences, and when input data from new projects for the purpose of evaluating Cost, Effort,

Table 3. Percentage of correct, false positive, and false negative results

Objective	Correct	False Positive	False Negative	Projects Considered
Effort	73.6 %	11.8 %	14.6 %	246
Cost	67.9 %	14.5 %	17.6 %	221
Quality	60.6 %	11.4 %	28.0 %	246

and product Quality, the framework predicts correctly 73.6% of the times effort related issues, 67.9% cost related issues, and 60.6% quality related issues.

5 Related Work

Overall, techniques in mining and reasoning in software repositories can be considered as falling into five main areas. The first area deals with statistical and data mining analysis of repository data to uncover statistically significant correlations or interesting trends as the software system evolves. In [15] data mining techniques are applied to revision history repositories to uncover dependencies between code segments that are difficult to extract with existing static and dynamic code analysis. In [16] Poisson modeling and generalized linear regression statistical analysis of change management data have been proposed as a way of predicting fault incidence in large long lived software systems. The second area deals with NLP type of analysis and the use of topic models for search and clustering such as Latent Semantic Indexing (LSI), Probabilistic LSI (PLSI) and variants of the Latent Dirichlet Allocation (LDA). In [17] a comprehensive survey of topic model based techniques for mining software repositories is presented. The survey provides a classificatory and comparative study of the different approaches found in the literature. In [18] a technique that is based on log reduction and Markov Logic diagnostic rules is used for root cause analysis. The main difference of this approach with the approach presented in this paper is that in [18] diagnostic rules are generated form plain AND/OR trees, while in this paper we present a framework that utilizes commitments, roles, and contributions for encoding Markov Logic Network rules from Goal Models. The third area deals with the extraction of metrics to compute maintainability indices, the identification of code cloning, and prediction of software quality. The fourth area deals with the analysis of software repositories using social network types of analysis. In [19] social network techniques are applied on repositories of email correspondents in order to address questions related to commit activities. In [20] a social network analysis is applied to reveal team communication patterns and assist on supporting management activities in software development projects. The fifth area deals with machine learning where predictions on specific software properties can be inferred by past data trends. More specifically, in [21] a technique that uses association rule mining and the k-Nearest-Neighbor machine learning

strategy to generate product-specific feature recommendations is presented. In [22], the authors discuss a technique for predicting latent software bugs that uses a machine learning classifier for determining whether a new software change is more similar to prior buggy changes or to clean changes.

In the area of probabilistic reasoning, GeNIe/SMILE [23] is focusing on decision making, that is to identify the best solution among alternatives and calculate an expected value based on a utility function and the probability associated with this solution. The GeNIe/SMILE framework utilizes influence diagrams as extensions of Bayesian networks to perform reasoning and assign probabilities for each possible outcome. GeNIe/SMILE and the goal model/MLN approach can both be considered as probabilistic reasoning methodologies. However, we believe that the goal models/MLN approach has two major advantages over the GeNIe/SMILE approach. First, goal models are more expressive than influence diagram as they allow for AND/OR logical operators to be used as well as a richer set of contributions compared to the simple influence arcs used in GeNIe/SMILE, and second they allow for the existence of cycles in the network the presence of which is a typical scenario when modeling complex interactions between decision items supporting a goal. Finally, MLNs provide an extension of the Bayesian networks in the sense that they combine the probabilistic reasoning capabilities of the Bayesian networks with expressive modeling capabilities of first order logic [8].

In [24] a policy verification framework is proposed. The framework uses the User Requirements Notation to model processes, and rules to denote policies. A bottom up linear propagation algorithm is used to compute the level of compliance of a parent node given key performance indicators and level of compliance of its children. The main difference from our approach is that our approach uses learning to calculate probabilities for each contribution link and utilizes a probabilistic reasoning method as opposed to a linear bottom up value propagation formula. Furthermore, with respect to model variability, the approach in [24] achieves variability (i.e. goal model families) by adding explicit OR children nodes denoting the variability conditions, while we use condition predicates on the contribution links of the model, and commitments to achieve variability.

In [25] an extension of the Goal-oriented Requirements Language is proposed by adding ranges of satisfaction values for each node. A satisfaction score propagation technique is then used to identify how new strategies may affect basis strategies and compute differences of satisfaction scores when alternative strategies are used. The main difference from our approach is that our approach does not focus on computing a difference in the satisfaction score between variance of a base strategy, but rather activate alternative models based on conditions that make contribution arcs true or false, and comments with reasoning for each alternative model independently.

6 Conclusion

This paper focuses in the area of software development analytics and in particular in the area of software project data analytics. The objective of this work is to

propose a qualitative framework, in which different stakeholders may state their goals, define how different views and roles may affect other goals and, allow for reasoning under uncertainty or partial information. Reasoning is achieved by the use of training a Markov Logic Network with past data and applying a Markov Logic reasoner. In this paper, we have shown how conditional contributions relate to roles and commitments, and we have defined the transformation from Goal Models and conditional contributions, to first order logic rules. In addition, we have discussed how these rules can generate a Markov Logic Network so that probabilistic reasoning can commence. The major contributions and novelty of this work is that it proposes a qualitative view of software analytics instead of a metrics-based quantitative one second, allows for information-rich goals to be defined capturing the different stakeholders views third, allows for valid and stable results to be reached even with partial data, a situation that often arises in the early stages of a project or on systems developed by different groups and involving different processes. The proposed technique has been applied with promising results to a repository of two hundred and fifty projects selected from the ISBSG portfolio of project data, pertaining to the top 25^{th} percentile of the largest projects measured by their Function Points. Future work in this area involves the compilation of goal models that relate to specific standards and processes (e.g. CMMI, SMART, SCRUM) denoting thus specific project management and organizational maturity views, and also the extension of the framework by allowing the definition not only of conditional contribution, but also of conditional decompositions and of conditional project objectives so as to increase the expressiveness of the models used.

Acknowledgment. This research is co-funded by the European Union (European Social Fund ESF) and Greek National funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund. This work is conducted in collaboration with IBM CAS Research. We thank the anonymous reviewers for their constructive and useful comments.

References

1. Menzies, T., Zimmermann, T.: Goldfish bowl panel: software development analytics. In: Proceedings of the 2012 International Conference on Software Engineering, ICSE 2012, pp. 1032–1033. IEEE Press, Piscataway (2012)
2. Buse, R.P., Zimmermann, T.: Analytics for software development. In: Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research, FoSER 2010, pp. 77–80. ACM, New York (2010)
3. Kwan, I., D.D.: A survey of techniques in software repository mining. Technical report, Software Engineering Global Interaction Laboratory, Univ. of Victoria
4. Ali, R., Dalpiaz, F., Giorgini, P.: A goal-based framework for contextual requirements modeling and analysis. Requir. Eng. 15(4), 439–458 (2010)
5. Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Reasoning about agents and protocols via goals and commitments. In: AAMAS, pp. 457–464 (2010)

6. Sebastiani, R., Giorgini, P., Mylopoulos, J.: Simple and minimum-cost satisfiability for goal models. In: Persson, A., Stirna, J. (eds.) CAiSE 2004. LNCS, vol. 3084, pp. 20–35. Springer, Heidelberg (2004)
7. Trendowicz, A., Heidrich, J., Münch, J., Ishigai, Y., Yokoyama, K., Kikuchi, N.: Development of a hybrid cost estimation model in an iterative manner. In: ICSE, pp. 331–340 (2006)
8. Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006)
9. Marc Sumner, P.D.: The alchemy tutorial, `http://alchemy.cs.washington.edu/tutorial/tutorial.pdf`
10. Lapouchnian, A., Mylopoulos, J.: Capturing contextual variability in *i** models. In: iStar, pp. 96–101 (2011)
11. Chopra, A.K., Dalpiaz, F., Giorgini, P., Mylopoulos, J.: Modeling and reasoning about service-oriented applications via goals and commitments. In: Pernici, B. (ed.) CAiSE 2010. LNCS, vol. 6051, pp. 113–128. Springer, Heidelberg (2010)
12. Giorgini, P., Mylopoulos, J., Nicchiarelli, E., Sebastiani, R.: Reasoning with goal models. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 167–181. Springer, Heidelberg (2002)
13. I.S.B.S.G.: ISBSG dataset release 11, international software benchmarking standards group, `http://www.isbsg.org/`
14. Jones, C.: Applied Software Measurement:Assuring Productivity and Quality. McGraw-Hill, New York (1991)
15. Ying, A., Murphy, G., Ng, R., Chu-Carroll, M.: Predicting source code changes by mining change history. IEEE Trans. on Soft. Eng. 30(9), 574–586 (2004)
16. Graves, T., Karr, A., Marron, J., Siy, H.: Predicting fault incidence using software change history. IEEE Trans. on Soft. Eng. 26(7), 653–661 (2000)
17. Thomas, S.W.: Mining software repositories with topic models. Technical Report 2012-586, School of Computing, Queen's University (2012)
18. Zawawy, H., Kontogiannis, K., Mylopoulos, J., Mankovskii, S.: Requirements-driven root cause analysis using markov logic networks. In: Ralyté, J., Franch, X., Brinkkemper, S., Wrycza, S. (eds.) CAiSE 2012. LNCS, vol. 7328, pp. 350–365. Springer, Heidelberg (2012)
19. Bird, C., Gourley, A., Devanbu, P., Gertz, M., Swaminathan, A.: Mining email social networks. In: Proceedings of the 2006 International Workshop on Mining Software Repositories, MSR 2006, pp. 137–143. ACM, New York (2006)
20. Wolf, T., Schröter, A., Damian, D., Panjer, L.D., Nguyen, T.H.D.: Mining task-based social networks to explore collaboration in software teams. IEEE Softw. 26(1), 58–66 (2009)
21. Dumitru, H., Gibiec, M., Hariri, N., Cleland-Huang, J., Mobasher, B., Castro-Herrera, C., Mirakhorli, M.: On-demand feature recommendations derived from mining public product descriptions. In: ICSE, pp. 181–190 (2011)
22. Kim, S., Whitehead, E., Zhang, Y.: Classifying software changes: Clean or buggy? IEEE Transactions on Software Engineering 34(2), 181–196 (2008)
23. GeNIe/SMILE, `http://genie.sis.pitt.edu/`
24. Shamsaei, A.: Indicator-based Policy Compliance of Business Processes. PhD thesis, University of Ottawa, Ontario, Canada (2012)
25. Amyot, D., Shamsaei, A., Kealey, J., Tremblay, E., Miga, A., Mussbacher, G., Alhaj, M., Tawhid, R., Braun, E., Cartwright, N.: Towards advanced goal model analysis with jUCMNav. In: Castano, S., Vassiliadis, P., Lakshmanan, L.V.S., Lee, M.L. (eds.) ER 2012 Workshops 2012. LNCS, vol. 7518, pp. 201–210. Springer, Heidelberg (2012)