ОЦЕНЯВАНЕ НА РИСКА ПРИ РАЗРАБОТВАНЕ НА СОФТУЕР ПО ГЪВКАВИ МЕТОДОЛОГИИ С ПОМОЩТА НА БАЙЕСОВИ МРЕЖИ

Людмила Димитрова[*], Кристина Петкова[**]
[*]*Университет "Проф. д-р Асен Златаров"* Катедра „Компютърни и информационни технологии", *8010 Бургас, България Бул "Проф. Якимов 1", e-mail: lyudim@gmail.com*
[**]*Технически университет-София , 1000 София, България, Катедра „Компютърни системи", бул."Кл. Охридски" № 8, e-mail: kristinapetkova.tu@gmail.com*

RISK ASSESSMENT IN AGILE SOFTWARE DEVELOPMENT USING BAYESIAN NETWORKS

Lyudmila Dimitrova[*], Kristina Petkowa[**]
[*]*University"Prof. As. Zlatarov", Bulgaria, Burgas 8010 Prof. Jakimov Str. 1, lyudim@gmail.com*
[**]*Technical University of Sofia, Bulgaria, Sofia 1000, Kl.Ohridski Str.8, kristinapetkova.tu@gmail.com*

ABSTRACT

Agile software development is an informal sum of methodologies and techniques for coordination of software projects with emphasis on close collaboration between the programming team and business experts. In this work, we suggest a Bayesian network model for risk assessment of the development of a high quality program by the established deadlines through probabilistic modeling of the processes specific for these methodologies. SamIam tool for modeling and reasoning with Bayesian networks is used.

Key words: *software development, risk of delay assessment, Bayesian networks*

Увод

Съществена цел в работата на всяка софтуерна фирма е създаването на качествен програмен продукт в установените срокове. Оценката и снижаването на рисковете на всички етапи от разработката е важен момент в управлението на проекта. Основните рискове при разработването на софтуер[1] са: принципиално трудната оценка на необходимото време, измененията в изискванията на клиента, движението на кадрите, недостатъците в спецификацията на заданието, недостатъчната продуктивност на началните етапи. Всички тези рискове са присъщи в пълна степен на проектите, разработвани по гъвкави методологии[2].

Един от широко използваните понастоящем методи за комплексна оценка на проектните рискове е използването на вероятностно моделиране с помощта на байесови мрежи[3]. Този подход е особено подходящ при анализ на рисковете, обусловени от човешкия фактор. Към тези процеси се отнася разботката на софтуер, в която проектните рискове са свързани с интелектуалната дейност на човека и влиянието върху нея на външни и вътрешни фактори.

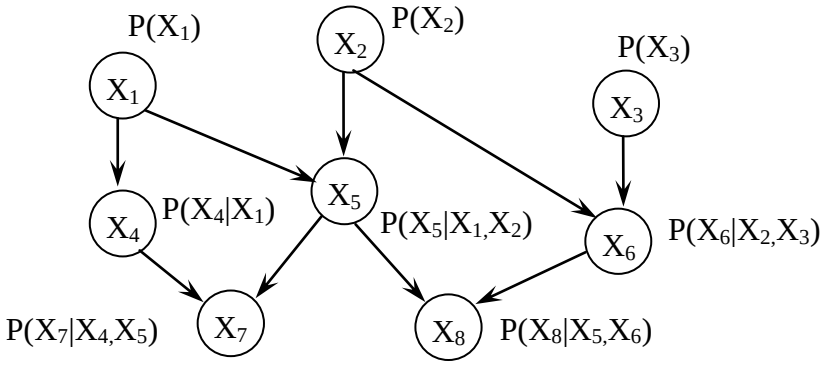
Байесови мрежи

Байесовите мрежи (Bayesian Networks, BN) обединяват принципите на теорията на графите, където сложна система може да бъде представена чрез по-малки и по-лесно управляеми части, и теорията на вероятностите, където тези части се комбинират последователно с цел ефективни логически разсъждения. По дефиниция [3] Байесовата мрежа е триплет N=(G,X,P), където:

- G=(V, E) е ацикличен ориентиран граф (directed acyclic graph, DAG) с възли V и ориентирани дъги E, задаващи взаимозависимостите между тях. Стрелките на дъгите задават посоката на влиянието, което обикновено е причина-следствие.

• $X=\{X_1, \ldots, X_n\}$ е множество случайни променливи, представляващи възлите на графа G. Всяка случайна променлива се характеризира с изчерпващо множество взаимноизключващи се състояния, често дискретни.

• P – множество разпределения на условните вероятности $P(X_i|pa(X_i))$ за всяка случайна променлива $X_i \in X$, където $pa(X_i)$ са променливите в „родителските" възли на X_i. За променливите без родители се използва маргиналното разпределение $P(X_i)$. За дискретните величини разпределението се задава чрез таблици на условните вероятности (conditional probability tables, CPT). Условните вероятности характеризират количествено влиянието на променливите-причини (родителските възли) върху дадената променлива.



Фиг.1. Проста Байесова мрежа

DAG използва понятието условна независимост (d-separation). Връзките, представени от дъгите, позволяват съвместното разпределение на вероятностите да бъде определено по израза:

$$P(X_1, X_2, \ldots X_N) = \prod_{i=1}^{n} P(X_i \mid pa(X_i)) \quad (1)$$

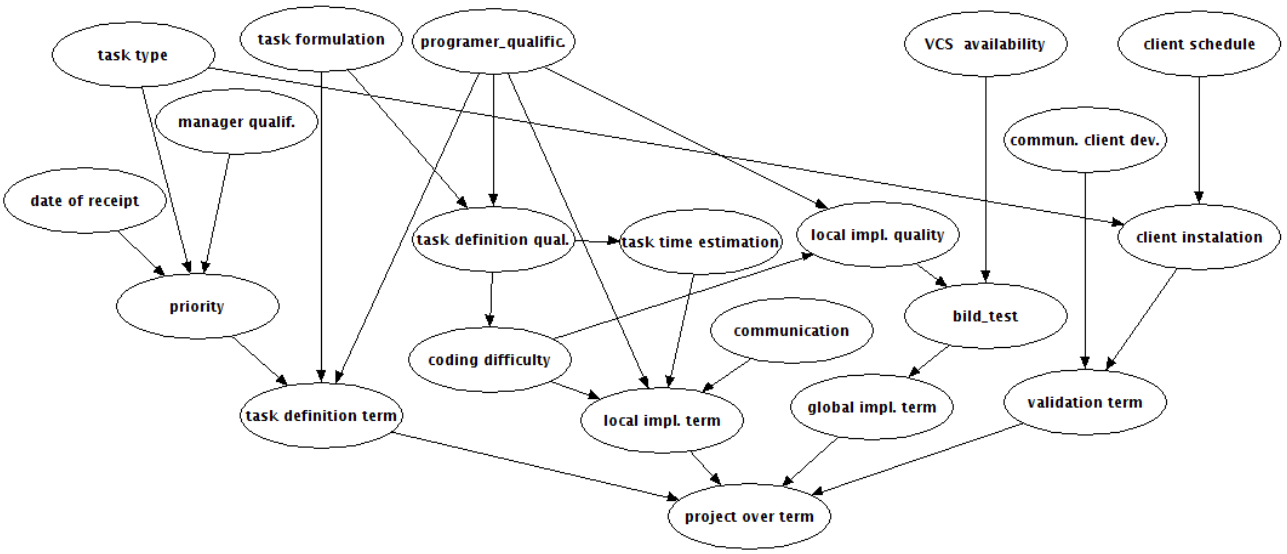
Например, за мрежата на фиг.1 съвместното разпределение на вероятностите може да се определи така:

$$P(X_1, X_2, \ldots, X_n)=P(X_1)P(X_2)\ P(X_3)P(X_4|X_1)P(X_5|X_1,X_2)\ P(X_6|X_2,X_3)\ P(X_7|X_4,X_5)\ P(X_8|X_5,X_6) \quad (2)$$

Една от базовите задачи на BN е разпространението на вероятностите – т.е. актуализацията на маргиналните верояности $P(X_i)$ след наблюдаването на значения на някои променливи.

Байесова мрежа за разглежданата задача

Забавянето в изпълнението на дадена задача и извънредната работа на програмистите с цел приключване на проекта в срок е често срещана ситуация при разработката на софтуер. Целта на предлаганата в настоящата работа Байесова мрежа е да представи взаимовръзките между факторите, които могат да предизвикат забавяне на проекта.

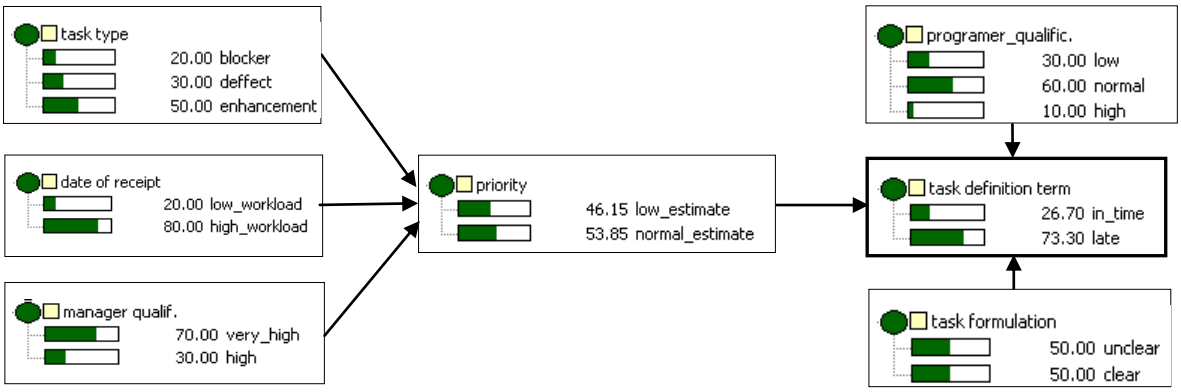


Фиг.2. Байесова мрежа в помощ на управлението на софтуерните разработки по гъвкави методологии

Първата стъпка в изграждането на мрежата е идентифицирането на фазите на разработка и степента на тяхното влияние върху своевременното изпълнение на проекта.

В предложената Байесова мрежа (фиг. 2) са отразени четирите основни фази на разработването на софтуерен продукт по гъвкави методологии: формулиране на задачата (*task definition term*), локална реализация (*local impl. term),* глобална реализация (*global impl. term*), и валидиране от клиента **(***validation term*). Забавянето на всеки един от тези етапи представлява рисков фактор за навременното завършване на задачата (*project over term*). За изброените 5 възела на мрежата са зададени по две възможни значения: своевременно приключване (*in_time*) и приключване със забавяне (*late*).

Преди да се започне работа по конкретна задача, тя трябва да се включи в процеса на планиране (фиг.3). Всяка една задача постъпва в опашката на определена дата, с описание и ниво на спешност. Променливата *date of receipt* определя работната натовареност на екипа в момента на постъпване на задачата и има две състояния: *low_workload* и *high_workload.* Променливата *task_type* е с три значения : *blocker, defect* и *enhancement.* С най-висок приоритет са задачите, които са проблемни за клиентите или блокират друг екип (тип *blocker*). Тези задачи се включват в процеса на планиране при първа възможност. Следващи по приоритет са дефектите от друг тип, открити от специалистите по качество или в отделите, използващи продукта (тип *defect*). С най-нисък приоритет са подобренията (тип *enhancement*). На базата на датата на получаване на задачата и нивото на спешност мениджърът приоритизира задачите. Степента на квалификация на мениджъра – възел *manager_qualif.* с възможни значения *high* и *very_high*, определя доколко точна ще бъде оценката му за приоритета – възел *priority* със значения *low_estimate* (приоритета, респ. времето за завършване на задачата е оценено по-ниско от реално необходимото), *normal_estimate* (реална оценка на приоритета) и *high_estimate* (завишена оценка).



Фиг.3. Оценка на риска на етапа на формулиране на задачата

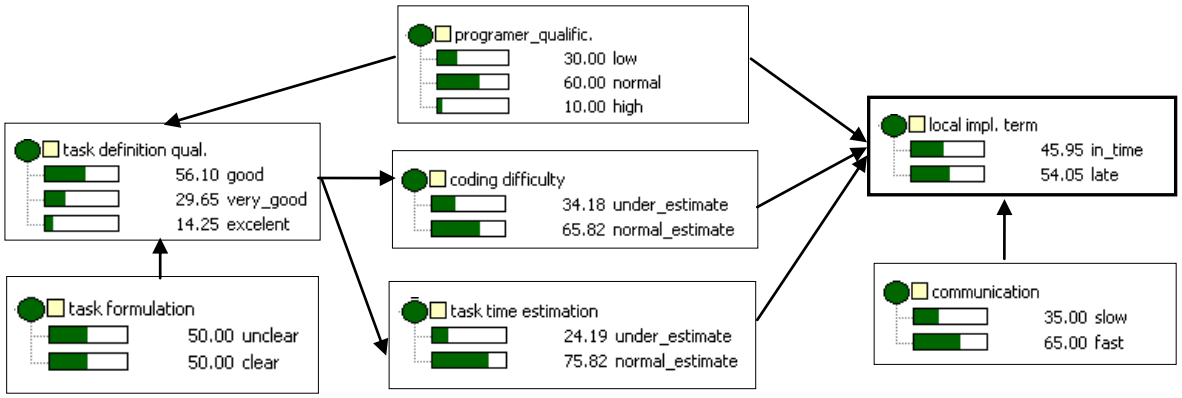
Основно значение за успешното завършване на задачата има нейното описание - възел *task formulation* със значения *clear* и *unclear*. Описанието представлява обяснение на проблема или подобрението, което се очаква да се направи. Освен текст то може да съдържа стъпки за възпроизвеждане, конкретни примери, скрийншоти, очаквани платформи, на които се среща или се очаква да работи. В случаите, когато описанието е непълно (*unclear*), работата на програмистите се затруднява много и забавя процеса. Те трябва да осъществят комуникация с лицето, което е отворило дефекта или подобрението, за да поискат повече подробности. Когато задачата е добре описана (*clear*), е по-лесно да се прецени колко време би отнела за разработване. По-точното планиране от своя страна помага за по-бързо завършване на задачата. Планирането на задачата се извършва от екипа на разработчиците. В зависимост от типа ѝ те преценят всички стъпки, които са нужни за завършването ѝ. Например при дефектите е нужно да се осигури време за имплементация, време за верификация от страна на специалистите по качество (тестване) и верификация от страна на клиента. Тази преценка е толкова по-правилна, колкото повече опит и квалификация има всеки от програмистите – възел *programmer qualific.* със значения *high, normal* и *low*.

На базата на приоритета, описанието на задачата и квалификацията на програмистите, се формира крайната дефиниция на задачата – т.н. дефиниция за завършеност. Колкото по-прецизно е направена тя, толкова по-реалистична ще бъде оценката на времето, нужно за завършването на задачата. Дефиницията на задачата като временен етап от решението ѝ се задава с обсъдения по-горе възел *task definition term,* а качеството на дефиницията за завършеност – с възела *task definition qual. (*значения *good, very_good* и *excellent).*

Крайната дефиниция на задачата се разглежда много детайлно при процеса на планиране. Въз основа на нея се прави преценка на сложността на проблема. Не винаги е възможно да се направи точка преценка - някои по-тривиални проблеми могат да бъдат оценени лесно, а когато проблемът е непознат за всички разработчици, като правило се предвижда допълнително време за проучването му. Този момент е представен с възела *coding difficulty* със значения *low_estimate, normal_estimate* и *high_estimate.*

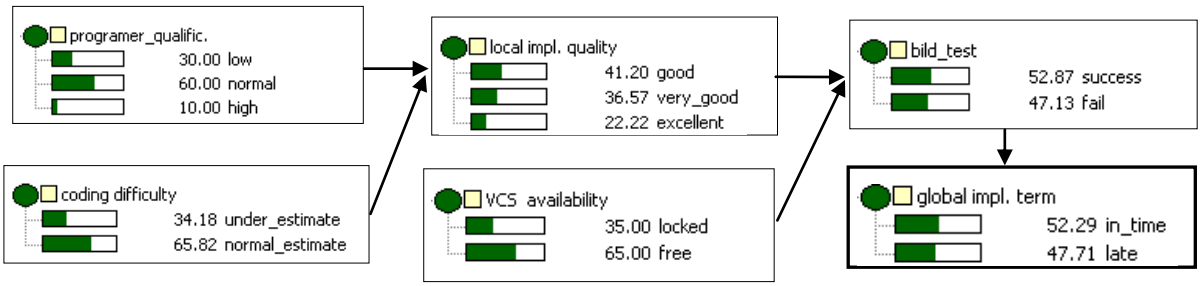
След като задачата е планирана за разработване и оценена, тя може да започне да се разработва. Според agile принципа всеки един програмист може да избере с коя задача да се заеме, независимо от квалификацията си. Комбинацията между сложността на задачата *(coding difficulty)* и способностите на разработчика *programmer qualific* определят колко време ще е нужно за завършване на примерното решение (прототипа) - т.е. дали локалната реализация (*local impl. term*) ще приключи в срок(фиг.4). Освен изброените две вероятностни величини времето за решение зависи от качеството на крайната дефиниция (*task definition qual.*) и от това дали на програмиста ще е нужна допълнителна комуникация за уточняването ѝ. Понастоящем много от големите фирми имат клонове в други страни. Разработчикът и

заявителят на задачата могат да се окажат в различни часови зони, с различни национални празници, което значително усложнява комуникацията между тях. Влиянието на комуникациите върху локалната реализация е отразено с възела *communication* (значения *fast* и *slow).* От друга страна, опитът на програмистите и сложността на проблема определят качеството на локалната реализция, т.е. доколко тя отговаря на нуждите на клиента – възел *local impl. quality* със значения *good, very_good* и *excellent.*



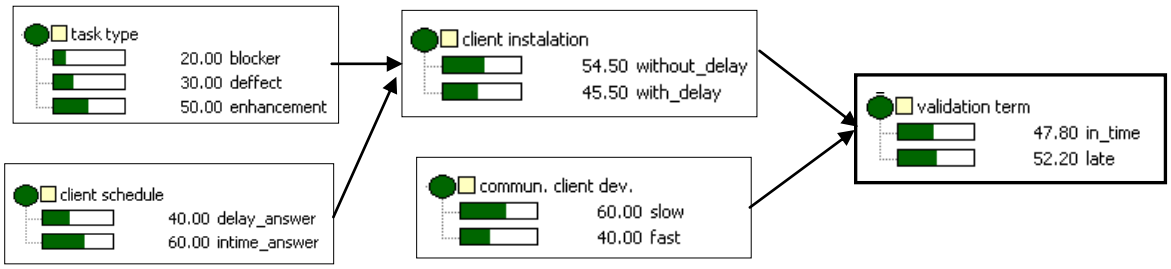
Фиг. 4. Оценка на риска на етапа на локална реализация на задачата

След завършване на примерното решение (*local implementation*) може да се премине към етапа на приобщеното решение (*global implementation*). Големите проекти често са интернационални. За да достигнат последните промени до всички, се използват системи за контрол на версиите – VCS. Състоянието на системата (*VCS availability* със значения *free* и *locked)* може да предизвика забавяне. Без примерното решение да е станало част от глобалната система, то не може да се тества за съвместимост с останалата част от продукта/продуктите. Възможно е да се открие несъвместимост между промяната и старата версия на кода. Тогава разработчикът трябва да поправи тази несъвместимост и да повтори процеса. Този момент е отразен с възела *bild test* (значения *success* и *fail*). Резултатът от тестовете на новия артефакт определя вероятността за приключване на приобщеното решение в срок (*възел global impl. term*-фиг.5).



Фиг. 5. Оценка на риска на етапа на глобална реализация на задачата

На последната фаза на разработката артефактът достига до заявителя на задачата, който трябва да го инсталира (възел *client installation* със значения *without_delay и with_delay).* Това може да стане бързо, но може да изисква цялостно обновяване на системата - в зависимост от вида на промените (*task type*). Забавяне може да възникне, ако заявителят има други приоритети (възел *client schedule* със значения *intime_answer* и *delay_answer*) или ако комуникацията между разработчика и заявителя е бавна (възел *commun. client dev.* със значения *fast* и *slow*). В случай, че проблемът е отстранен, т.е. глобалната имплементация работи, заявителят валидира, че задачата е завършена (възел *validation term*-фиг.6).



Фиг. 6. Оценка на риска на етапа на валидиране на задачата

С помощта на предложения модел може да се прави динамична оценка на хода на разработката след постъване на дадени наблюдения. Например, при наблюденията *task type=defect, task formulation=unclear, date of receipt=high_workload, programmer qualific.=low,* изчислената вероятност за приключване на проекта в определения срок е 23%. Това означава, че трябва да се включат допълнителни ресурси в разработката и/или да се въведе резерв от време за преодоляване на риска от закъснение.

Изводи

Предложена е Байесова мрежа, с помощта на която може да се изследва влиянието на отделните фактори върху процеса на завършване в срок на проект, разработван по гъвкави методологии. Параметрите на мрежата могат да се зададат на базата на опита и/или чрез обработка на наличните данни. След тестването ѝ тя може да се използва за анализ и прогнозиране на различните ситуации при работа по аналогични проекти.

Литература

1. DeMarko, Tom, Timothy Lister, 2003, "Waltzing With Bears: Managing Risk on Software Projects", Dorset House
2. Martin, Robert C., "Agile Software Development, Principles, Patterns, and Practices", Prentice Hall 2002
3. Pearl, J., 2009. Causality: Models, Reasoning and Inference, second ed. Cambridge University Press.
4. Fan, Chin-Feng, Yuan-Chang Yu, "BBN-based software project risk management", The Journal of Systems and Software 73 (2004) 193–203