Quantitative Software Engineering

Igor Schagaev, London Metropolitan University, i.schagaev@londonmet.ac.uk
Svetlana Anulova, Ins of Problem Control, Russian Academy of Sciences
Hamid R. Arabnia, The University of Georgia, USA

Abstract

This paper proposes an analysis of a project model with feedbacks as palliative to qualitative models widely utilized in software engineering. It demonstrates that software engineering projects can reliably be modeled and analyzed quantitatively. Several analytical methods are introduced aiming to resolve problems of quantification. A sequence of steps to introduce rigor in software engineering models is explained showing how software warehouses can adopt proposed methods. Proposed approach when implemented can enhance the quality of produced software.

Keywords

Software engineering, qualitative models, feedbacks, quantitative modeling, simulation, semi-Markov models, analytic tractability.

1. Introduction

Software systems developed using ICT has become extremely complex and very often unmanageable. The problems of design and further maintenance have become a significant challenge for developers; in particular, in safety critical systems, including aviation and ground transport, pipelines, and others. The tangible cost associated with corrective measures is often extremely high.

For example, in recent years Chrysler [1],[2], Toyota [3], and other automobile manufacturers were forced to re-call hundreds of thousands of their new vehicles for corrective technological flaws that were not detected during manufacturing.

Due to various technological flaws, some corporations are even considering discontinuing some of their products; for example, Airbus is considering stopping the production of their A-380 [4]. There are many other examples of corporations discontinuing products solely due to costs associated with corrective measures and maintenance. It should be noted that a high percentage of cost of many products are due to software and electronics that operate the actual products. For example, 60% of cost of building aircrafts is due to ICT systems and the associated software.

Inherent in the design and creation of most systems and projects are User, Hardware, and Software. Obviously, any fault and flaws of one is intertwined with others. Thus projects with classic phases of concept, design, and development are becoming processes and phases of states that are tightly coupled in all steps of their development.

The challenge to control, manage, and streamline such processes have become an immense challenge. There are many potential solutions, based on methodologies, assumptions, methods to apply and follow but none of them considered to be solutions that would address the problem as a whole and estimate, let say, an impact of phases of the project on each other, values or weights of project phase dependencies or timing of decision making.

The software engineering researchers and community are publishing manuals [6], [7], new methodologies and ontologies (Agile, etc.) [8]. Many top tier international conferences, workshops, and symposiums report possible solutions every year. Unfortunately, most reported solutions, though important contributions, only address specific problems in isolation to other problems. Regretfully, there is no breakthrough on the subject. One of the major reasons is qualitative analysis and qualitative methodologies used.

This paper attempts to provide a scheme of redesign of software engineering models that in the long run would provide a quantitative measure. This paper also attempts to highlight problems of “quantification” and briefly present some solutions or pathways to find solutions.
2. Problem of uncertainty and qualitative approaches

The model of project process with feedbacks was introduced in [9] and first attempt of its quantification was introduced using simulation further developed in [10],[11]. Recent development [12] and two mentioned previous were based on various techniques of numerical solutions.

Project process as Markov process was introduced more than half a century ago by P. Howard [13], attempting to analyze project process analytically.

The costs of project phases were never certain, “generalization” of project phases or work packages independency failed dramatically even at the level of elementary projects, forcing us to accept that phase interdependencies are required to take into account. Therefore, managing our projects within budget and time constrains we have to handle project phases dependencies and handle them effectively.

Project complexity, primarily software development projects, growths and only increase the need develop some tangible solutions.

Indeed, having a quick look at the Figure 1 one might observe that phases of the project (states 1 to 3, and 4 is completion state) are dependent, and some redo takes place.

![Figure 1 Project phases with full feedbacks](image)

Immediate attempt to resolve this uncertainty was and still is based on formation of system of differential equations with assumption of Markov or semi-Markov properties of the states and transitions between them.

Regretfully, turning back to Fig.1 we can find that number of unknown variables (all feedback links from states 3,2,1 including self-feedbacks) exceeds number of equations, defined by number of project states. Thus the first option to solve it based on numerical solution, one of them is our own attempt, as it was presented in [10],[11].

Unfortunately, if we follow the same pathway, instead of solving project of process problems of improving efficiency, reliability and handling the cost of project we will be addressing mathematical complexities instead of project ones.

There is no doubt that analytical solution of process of project flow equations is preferable. This way we are able to determine dependencies between phases and learn how to cope with overheads of time or cost.

3. Existing and proposed solutions

There is no doubt, the less we use “guestimations”, math methods assumptions to find a quantitative solution the better. We should avoid a situation when complexity of model exceeds complexity of a process we try to control.

Using [14], further developed [15],[16],[17] we might find that three redundancy types can be applied separately or in combination to reduce uncertainties existing in the description of the project as a model. They are:

- **Structural (S)**, **Informational (I)**, **Time (T)**.

**Structural redundancy** might be considered when we add new equations that reduce number of unknown variables.

**Informational redundancy** might be introduced as extra knowledge of behavior of the system of equations – for example number of iterations – i.e. how many times we have been visiting previous states, or profiles of values of feedbacks;

**Time redundancy** might be considered as introduced longer period of system observation.

Thus, introducing in the description of the system more equations, or more knowledge on behavior of unknown variables or observing this process longer enable us to find analytical solution.

One of the relatively recently developed models is presented in the next section.
4. Project as system with feedbacks and rewards dependent on past

R.A. Howard [13] introduced Markov chain with rewards to described economic objects and analysis of strategies for their control. Model became useful, a lot of research were taken, with introduction of generalization and special features.

Up to now there were no researches of Markov model when gain depends on the number of iterations of revisit projects states, and analysis of a reward scheme. Indeed, we are not redoing fully the project phase (states and Fig.1), but we simply must accept that further revisits might be reduced in accompanied cost.

This scheme suites well to software development project life cycle with feedbacks, where we assume that we are re-doing some bits of previous phases of project when errors or incompleteness of previous phases were detected.

The same models became useful for wider domains of human activities, including systems that include information and computer technologies, and in particular large-scale software projects.

Naturally, modeling of re-doing some steps of the project – should be rather typical case in everyday project practice. Surprisingly, this kind of model was not developed and missing so far.

Thus, as a first attempt we have to consider processes when gain depends on number of iterations of each phase of the project, perhaps with assumption of some sort of discounting factors.

Next section presents one of possible method of calculation of the project result (we use term reward to indicate that project cost might be too high) with assumptions of counting number of iterations and discounting of reward or cost of redoing.

Problem

Consider Markov chain with states \( \{0, \ldots, K\} \) and transition \( \{ P = p_{ij}, i, j = 0, \ldots, K\} \) and absorbing state \( K \).

State of Markov chain at the moment \( n = 0, 1, \ldots \) denote \( x(n) \).

Set trajectory \( x = (x(0), x(1), \ldots) \) for the state \( i \in \{0, \ldots, K\} \).

Denote as \( \phi_i(m) \) number of visits of the state \( i \) during period \( [0, 1, \ldots, m] \), then

\[
\phi_i(m) = \sum_{n=0}^{m} I_i(x(n)), \quad \text{and} \quad \phi(m) = (\phi_1(m), \ldots, \phi_K(m))
\]

For \( N = 0, 1, \ldots, \infty \) consider random value

\[
\xi(N) = \sum_{n=0}^{N} f(x(n), \phi(n)),
\]

where

\[
f(i, \phi) = \frac{K-i}{1 + \max_{j \neq i} \phi_j}.
\]

We attempt to derive \( E_\xi \xi(N) \), including \( E_\xi \xi(\infty) \)

First impression that this scheme is similar to discrete Markov chain with reward [18]. But this is not.

New approach to calculation of \( \xi(N) \) assumes extension of phase space \( \{0, \ldots, K\} \) and, therefore process \( x \) in a way that enables to create Markov chain and function \( f \) as a function of its own state only.

Then it becomes possible to apply recurrent scheme of first segment of Chapter 2 [13], travelling in simplex

\[
\{ \phi \in [0, 1, \ldots)^K : \sum_{i=1}^{K} \phi_i \leq N \}
\]

along the layers: from set

\[
\left\{ \sum_{i=1}^{K} \phi_i = n \right\}
\]

to set

\[
\left\{ \sum_{i=1}^{K} \phi_i = n - 1 \right\}
\]
(These layers play a role in our case that in classic problem of [13] plays time).

At the moment we are not concentrating on this similarity, because our goal is to investigate cases when mean time of request in the network (chain) is possible to calculate analytically.

Case I: discounting accordingly number of visits

If \( f(i, \phi) = f(i) e^{-c_i \phi} \)

Where \( n_i \) - vector with non-negative coordinates in \( \{c_i(0), c_i(1), \ldots, c_i(K)\} \), then classic scheme of Markov chains with rewards suits [18]. This becomes obvious if we note that

\[
\{c_i, \phi(N)\} = \sum_{n=0}^{N} c_i(x(n)).
\]

**Theorem 1.** Function \( v: \{0, 1, \ldots, K\} \rightarrow [0, \infty) \), \( v(i) = E_i \xi(\infty) \)

is defined by system of linear equations

\[
v(K) = 0, \quad v(i) = e^{-c_i(i)} \left( f(i) + \sum_{j \neq i} p_{ij} v(j) \right).
\]

Theorem enables immediate generalisation for the case when reward consists of sum of rewards as above, for example,

\[
\sum_{i=1}^{K} \sum_{n=0}^{N} f_i(x(n)) \exp \left\{-\sum_{m=0}^{n} c_i(x(m))\right\}.
\]

Case II Function of reward with two variables

Consider a simplified problem: reward that received in a state \( i \in \{0, \ldots, K\} \), depends not on the whole vector \( \phi \), but only on \( \phi_i \) - number of visit (iterations) of this state. This means that function \( f \) depends on smaller number of variables:

\[
f: \{0,1,\ldots,K\} \times \{0,1,\ldots\} \rightarrow [0,\infty).
\]

The idea here is to exploit a property of linearity of \( \xi_i(\phi) \) by \( f \).

Let \( f \) differ to zero only at the state \( i \in \{0, \ldots, K\} \), in other words \( f(j,l) = 0 \) at \( j \neq i \). Then

\[
E_i \xi(N) = \sum_{n=0}^{N} P_i(n \leq \phi_i(N)) f(i,n)
\]

Value \( P_i(n \leq \phi_i(N)) \) depends only on transitional probabilities and when \( N \rightarrow \infty \) goes to \( p^n_i \), where \( p_i \) denotes probability of the fact that chain leaving node \( i \), returns to this node once more time. To define rigorously this probability, we introduce for \( i \in \{0, \ldots, K-1\} \)

a random variable \( \tau_i \) - a Markov moment of chain in the state \( i \)

\[
\tau_i = \min\{n \in \{1,2,\ldots\}: x(n) = i\} \quad (\min\{\emptyset\} = \infty)
\]

(moment of the first visit after 0 iteration for the state \( i \)).

If Markov chain at moment is not in the state \( i \), then \( \tau_i \) coincident with moment of reaching the state \( i \); while when process is in the state \( i \) then \( \tau_i \) is a moment of first return to the state \( i \).

Let \( p_i = P_i(\tau_i \leq \tau_K) \)

Then,

\[
E_i \xi(\infty) = \sum_{n=0}^{\infty} p^n_i f(i,n).
\]

Further, for \( j \in \{0, \ldots, K-1\} \)

\[
E_j \xi(\infty) = P_j(\tau_j \leq \tau_K)E_j \xi(\infty).
\]

**Theorem 2.** Let \( i \in \{0, \ldots, K-1\} \) and function \( v: \{0,\ldots,K\} \rightarrow [0,1] \) match system of linear equations

\[
v(j) = \sum_{l=0}^{K} p_{jl} v(l), \quad j \neq i,
\]

\[
v(i) = 1, \quad v(K) = 0.
\]
Then

\[ p_i = p_{i0} + \sum_{j=0, \ldots, K} p_j v_i(j). \]

Proof is based on the well-known fact: value \( v_i(j) \) – is a probability of the achieving the state \( i \) from state \( j \) earlier that in state \( k \),

\[ j \in \{0, \ldots, K-1\}. \]

These two examples illustrate that analytic solution of possible when we introduce extra information about system behavior – number of iterations and rate of discount assumed along the process.

5. Next steps

Analytic modeling of project flow offers two areas of further research: a) regarding mathematical approaches – what else is possible to apply making analytics and b) which way analytic results might be applied – i.e. how to use the results of this analysis?

5.1 In math

Another missing point in creation of Markov model for project process is behavior of feedbacks. Indeed, an assumption of neat execution of phase after phase for any project is unrealistic; we have to assume non-zero probabilities of transition backward to previous phases.

Semantic of feedbacks provides some helpful dynamic constrain that ease analytic solution. It based on assumption of value of feedbacks. Clear, that the last thing project manager wants is total redoing of all phases of the project from the final step.

In turn, our assumption on non-zero values of feedbacks such as self-feedback – redoing of the same phase, or redoing of some elements of previous phases is much more natural.

Thus we might introduce an extra information about behavior of feedbacks assuming that probability of longer feedbacks is smaller than short ones and defined by, for example, a Poisson distribution. This will keep further solutions within reach of analytic method.

This form of information redundancy - the known form of ratio of feedbacks might be really useful. The use of time redundancy to resolve uncertainty of feedback values in our case present much less value in finding of project parameters.

5.2 In application for SE

Application of either first introduced here model - with counted feedbacks or the second mentioned - with information redundancy fine-tuning of feedbacks as a function of their length raises the following questions:

- How can we apply analytic methods of project evaluation in practice of software projects?
- How can we apply the model of a project integrated with other projects within company?
- Can we apply integrated model of several projects as a single entity, using it at the level of corporation?

The sequence below (Fig.2) presents one of the possible approaches.

| Extract financial data for the project from corporate account |
| Relate project phases with financial data available |
| Introduce expected costs per project phase |
| Introduce conditions for finding values of feedbacks: |
| - Numerical, such as in [10,11] |
| - Discounted feedbacks such as section 4 |
| - Discounted feedback as a function of length section 5.4 |
| Define for team and technology: |
| Cost of retraining and/or upgrade required |
| Estimate potential gain from solutions proposed: project-wise, corporate-wise |

Figure 2 Implementation of QSE

At first, from existing financial data about project and project phases we have to extract the values of project phase planned cost and project phase.

Next we have to create a scheme, similar to Fig.1, introducing feedbacks between phases. Further – from corporate data generalized expected cost and time should be introduced.

Solving equations either with discounted returns or with introduction of distribution of feedbacks
as mentioned above we can estimate realistically project cost of concrete project and apply the scheme for similar projects.

For serious projects and corporate programs and state-size programs a customer or state representative (or - internally quality control analysts) might be able to provide values of feedbacks.

Having feedbacks values we can derive which distribution they obey. Having “golden” standard” for company data about projects – developers call it BEP – best existing practice our model enables to check level of competence and efficiency of project team in “the small” - per work package or independent task and “at large”, taking into account dependencies of work packages and their impacts on each other.

We did apply this approach for international projects with respectable Seattle-based companies and several companies from various European countries. Very strange behavior of feedbacks was detected and this result was used as evidence of weaknesses of some teams or technologies applied along the phases of large-scale project.

Corporate-wise application is also possible, again by tuning a model for corporation or industry typical projects. Derived from this exercise pattern might helps to create an efficient working model. This way we might use a model to create a software tool to evaluate efficiency of corporate operations. Having this analysis a corporation might objectively assess when technology or competence of project teams is becoming obsolete and need upgrading.

6. Conclusion

- An attempt to introduce analytic schemes to quantify project process flow with realistic assumptions of overlapping phases is introduced.

- Shown that taking into account feedbacks between phases of the project makes model of the project realistic;

Ways to apply analytic solutions for the model of project with feedbacks are proposed and explained, resolving uncertainties along project progress.

- Further work of the quantification of software engineering and similar project control is possible in development of model with discounted feedbacks as function of their length;

- Proposed analysis and methods might become a core of application tools or software framework to enable project engineers and managers to analyze impact of their actions in advance.

References

[8] Meyer B. Agile! The Good, the Hype and the Ugly. ISBN 978-3-319-05155-0