MANAGING AGRICULTURE SOFTWARE PROJECTS IN THE NEW ECONOMY*

R. Pergl

Czech University of Life Sciences, Faculty of Economics and Management, Department of Information Engineering, Prague, Czech Republic

This paper presents an original method for identifying needs for changing a software project structure. The method is based on the theoretical background of systems theory and modelling and complex adaptive systems.

software engineering; agriculture information systems development; managing software projects; systems modelling; complex adaptive systems

INTRODUCTION

The New Economy brings high demands on technology and communication (K e l l y , 1999). Moreover, agriculture information systems tend to be more and more complex. The result is that the development of information and knowledge systems becomes extremely demanding, because:

- The structure of information stored in systems is complex, polymorphic with many relations and exceptions.
- Requirements are driven by the marketplace, which changes very rapidly.
- Requirements may not be clear in the beginning of the project.
- The number of participants involved in system development and utilization is increasing.
- Heavier requirements on security and reliability.

There are a lot of methodologies for managing software projects. The problem is that it is not easy to decide which methodology is the right one for a particular project. And even under one methodology, there are typically many possibilities of how to set the infrastructure and processes. In current practice, experience, advice and intuition are used. A high number of failing projects (S t a n d i s h , 1995) shows, that this may not be always enough.

One of the most crucial keys to success is how to get from the requirements phase of the proposed software project to an ideal set-up for the software project infrastructure and processes. This transformation should have the following attributes:

- structured, so that the transformation may be comprehended,
- documentable, so that conclusions can be verified and/or used for future projects,
- traceable, so that conclusions may be audited.

This paper presents an original method based on the systems theory and modelling. Project management approach is influenced by the complex adaptive systems.

MATERIAL AND METHODS

The method presented here is based on the analogy between systems theory and software project management. A formal model of a system consists generally of inputs, outputs, inner elements and relations (S k y t t n e r , 2001). Inputs are divided into endogenic ones, which are inputs crucial for the system model and exogenic, which are other inputs that must be taken into account. The analogy between the general system model and a software engineering project is shown in the Table 1.

Inner elements may be, according to the concrete purpose and goal of the model:

- objects,
- classes.

Objects are chosen in the case where it is necessary to model the system in a detailed level of single objects: documents, team roles, etc. In general methodological models, classes will be usually used. Each element then represents a whole class, not an individual object: e.g. “programmer” represents all the programmers. We do not care about structure and dynamics of objects inside the class. For the purpose of this paper, we will assume that all the inner elements are classes.

Now we can speak about *software project management* from the perspective of systems modelling. Inputs and outputs are given, so we may manage:

- inner elements,
- relations between inner elements,
- relations from inside to outside.

Managing the software project thus means managing those elements. In reality, they may be connected into indivisible structures. Let’s define that a **project factor** is

* Supported by the Ministry of Education, Youth and Sports of the Czech Republic (Grant No. MSM 6046070904 – Information and knowledge support of strategic control and No. 2C06004 – Information and knowledge management – IZMAN).

Table 1. Analogy between the general system model and a software engineering project

System modelling term	Software project analogy	Set symbol
Endogenic inputs (crucial inputs interesting for the modelling)	explicit software product requirements	I_S
Exogenic inputs (other inputs)	external conditions both predictable and unpredictable (environment)	I_E
Inputs (union of endogenic and exogenic inputs)	all external factors influencing the project	$I = I_S \cup I_E$
Outputs	• software product and its parametres • technological environment for running the product • documentation and other artefacts • training	O
Inner elements	• team (project roles) • subcontractors • tools (both development and supporting) • artefacts (code and documentation)	P
Inner relations (relations between inner elements)	• process • project management • intra-team communication • subcontractor communication	R_I
Inner to outer relations	information to the customer	R_{IO}
Outer to inner relations	information about the requirements changes	R_{OI}
Relations from inside to both outside and inside	cooperation requests to the customer from the team	R_{IOI}
Relations from outside to both inside and outside	the team responds to immediate customer requests for cooperation	R_{OIO}
Relations (union)	all relations	$R = R_{OIO} \cup R_{IOI} \cup R_{IOI} \cup \cup R_{OI} \cup R_{OI} \cup R_{IO} \cup R_{IO} \cup R_I \cup R_I$

1. An inner element.
2. A relation between inner elements.
3. A relations from inside to outside.
4. Any sub graph of a graph consisting of a set of nodes P and a set of edges $R_I \cup R_{IO.}$ The set of project factors will be marked C.

If we suppose that the goal of a project is to achieve a relation between inputs and outputs, then project management means fine-tuning the project factors. This happens based on the inputs and represents an *adaptation of the system* according to inputs in such a way that the system achieves its goal in an optimal way, i.e. with minimum resources (time and finance).

An important problem is detecting the need for adaptation. There are two possibilities:

1. *Adaptation ex post*, that is based on past. The adaptation driver is discrepancy between outputs and inputs. This type of adaptation may be used with iterative development life-cycle.
2. *Adaptation ex ante*, that is considering the future. This type of adaptation is performed based on prediction about the needs of structure changes.

We observe that these two types of adaptations are usually mixed in the practice. For managing software projects, the second type is very important. Adaptation based on prediction prevents problems and mitigates risks.

The following contribution describes a method for effective ex ante adaptation management.

RESULTS

The method

For implementing a practical tool, it is necessary to select certain relation types to achieve a compromise between model accuracy and ease of use. Because in practice, the requirements on the software product often prevail above stochastic inputs, the selected relation represents *demands* on the system. Formally, let us define:

Demand dem(a, s) of the input a to the project factor s, where $a \in I$ a $s \in C$. The domain range is an ordinal scale <0, 10>. 0 means, that the input a does not require an adaptation of the factor s. The higher the value, the higher the adaptation that is needed.

An example may be the situation where the project leader comes to the conclusion that because of an input representing product portability, factors related to software platform portability will have to be adapted.

As for the relations between the inner elements, substitutability is one of the most important. Demands for adaptation of one factor may be mitigated by the substitution of another factor. An example may be providing training to team members instead of hiring a new needed expert role. Substitutability is defined as:

Substitution of project factors s_1 and s_2 sub(s_1, s_2), where $s_1, s_2 \in C$, is a mapping $C \times C$ onto an ordinal scale <0, 10>. The substitution represents the possibility of sub-

stituting a demand on project factor s_1 by a substitute s_2. In the case where substitution is not possible, the function has a value of 0. The higher value, the higher possibility of substitution. The value 10 means perfect substitution.

Individual demands on factor adaptation are added and we get the total demand. This total demand will be called the difference:

Difference of the factor s_j, where $j = 1, \ldots, n$ is the value of the function $\mathrm{dif}(s_j)$, that assigns a non-negative whole number of every project factor $s_j \in C$:

$$\mathrm{dif}(s_j) = \sum_{i=1}^{m} \mathrm{dem}(a_i, s_j)$$

where a_i is input, n is the number of project factors and m is the number of inputs.

One factor may be substituted by multiple substitutes, as covered by the following definition:

Total substitution of the project factor s_j, where $j = 1, \ldots, m$ is the value of the function $\mathrm{csub}(s_j)$, that assigns a non-negative whole number to every project factor $s_j \in C$:

$$\mathrm{csub}(s_j) = \sum_{k=1, k \neq j}^{n} \mathrm{sub}(s_j, s_k)$$

where n is the number of project factors.

The resulting demands on factor adaptation may be thus mitigated by inner substitution relations. We get the resulting difference of the factor:

Resulting difference of the project factor s_j is the function

$$\mathrm{vdif}(b_j) = \max(0, \mathrm{dif}(b_j) - \mathrm{csub}(b_j))$$

The resulting difference represents overall netto demands on factor adaption. The factors with highest values are the most crucial topics for project management.

Inputs and factors selection

The next step is to select appropriate inputs and project factors. The sets I and C are in reality very large. For practical applications it is necessary to specify a subset of the inputs $I_2 \subset I$ and a subset of the project factors $C_2 \subset C$. Ideal attributes of those sets should be:

- completeness,
- independence,
- minimalism.

In practice, it is very hard to achieve perfection in all those parameters and we make a compromise between completeness and model comprehensibility and manageability, because the time complexity of processing all demands according to the definition is $\Theta(|I_2| \times |C_2|)$.

Using the method

The process of evaluating the resulting differences is as follows:

1. *Requirements gathering*. Requirements gathering by classical methods (interviews, questionairres, etc.).
2. *Structuring requirements*. Informal requirements are transformed to method's system inputs.
3. *Requirements analysis*. This step means identification and quantification of demand functions. For all pairs of inputs and factors, we analyse whether the input demands some sort of adaptation. For factors not present, the adaptation means the adoption of this factor. The demand then represents the complexity of the factor implementation.
4. *Difference function evaluation* according to the formula above.
5. *Substitution functions and total substitutions evaluation*. For factors with high differences, the high adaption demand may be mitigated by identifying some substitution relations. Substitutions for each factor are then summed according to the formula above.
6. *Resulting differences evaluation* according to the formula above.
7. *Results interpretation*. Non-zero resulting differences show the needs of factor adaptation. The higher the value, the higher the needs of overall adaptation. It is necessary to note that the resulting difference has an absolute quantitative character while the demands may have various qualitative characters as well. Those qualitative characters cannot be easily expressed in the method. Thus situations may occur when two demands on an adaptation may go even against each other. The resulting difference represents just a sum of all the demands and its high value must be interpreted correctly according to the reality of the situation. Appropriate management actions must be undertaken then. Sometimes, no action may be the best action.

The described steps are performed in the preparation phase but may be put more precisely later. The first preliminary evaluation of adaptation needs is performed after the first round of high-level requirements.

A practical example

Let us show one small practical example demonstrating the use of the method. Let us imagine that we need to develop an information system for the cattle farm. The system should hold the evidence of the cattle, the evidence of the veterinary inspections and lactation evidence.

The first task is to select appropriate inputs and project factors. As inputs I_2 we may choose quality characteristics according to ISO/IEC 9126-1. The norm defines six characteristics:

- functionality,
- reliability,
- usability,
- efficiency,
- maintainability,
- portability.

As for the project factors C_2, let us focus on the categories: team characteristics, roles and process and let us specify the following factors:

- team characteristics
  - qualification,
  - personal stability,
  - personal commitment,
- roles:
  - team leader,
  - analyst,
  - developer,
  - tester,
  - technical writer,
  - subject matter expert.
- process:
  - development process flexibility,
  - risk management,
  - quality assurance.

The next step is valuing the demand functions. The valuation is based on interviews and information gathering about the needs of the future system. We may, for example, learn that the system must be very *reliable* and this makes demands on our *team qualification*, on the *tester role* and also makes the *quality assurance* process a crucial one. There are a few functions that will need some more qualification, but the worst issue is that the project is large and complex and there is not much time. It makes demands on the team *commitment*. Unfortunately, it looks like the team commitment is not high and should be increased, and fortunately, the team is at least *stable* and the personality of the *team leader* is a great example for the team members. Users demand the possibility of remote lactation data gathering. This will be solved by porting the solution to mobile devices. This *portability* will require more *team qualification* and will enhance demands on the *tester role* and overall *quality assurance*.

First, we fill in the demands table (Table 2). Only rows and columns with at least one non-zero demand are shown.

Next, we quantify the substitution functions like:
sub(commitment, personal stability) = 2
sub(commitment, team leader role) = 3

By incorporating these substitutions we obtain a table with resulting differences (Table 3).

Now we can interpret the result. The analysis shows us, that the most crucial areas that will require adaptation are team qualification and quality assurance. The tester role will have more demands which can be fulfilled either by hiring another tester or increasing the tester's load. Increased personal commitment will be required, but it will be mostly mitigated by substitutes, so there is no big need for adaptations.

DISCUSSION

Large agriculture software projects tend to be very complex and demanding. Project management is a sort of balance between control with rigour and coaching with creativity. Demands on accurate cost and time estimations force detailed planning and adherence to the plan. On the other hand, it limits the flexibility, which is crucial in today's turbulent environment of changing laws, regulations, globalization and other aspects. Nevertheless, more flexibility, creativity and less control requires high level of experience and expertise and leads to risks that the project will run out of control and will not fulfil its purpose.

Various methodologies address various risks of project failure. The so-called "rigorous methodologies" focus on processes and planning. An example of rigorous methodology is Rational Unified Process (RUP) that is widely used. On the other hand we have the family of "agile methodologies" that rely more on people and commitment. We can name Extreme Programming, Scrum, Adaptive Software Development, Dynamic Software Development Method and others.

All agile methodologies honour the following principles and practices:

Table 2. Demands analysis

Dem(*a*, *s*)	**Inputs *a***				**dif(*s*)**
		reliability	functionality	portability	
Factors *s*	team qualification	6	2	5	13
	commitment	0	8	0	8
	tester role	3	0	8	11
	quality assurance	8	0	5	13

Table 3. Resulting differences

		Total difference dif(*s*)	Total substitution csub(*s*)	Resulting difference vdif(*s*)
Factors *s*	team qualification	13	0	13
	commitment	8	5	3
	tester role	11	0	11
	quality assurance	13	0	13

- **Customer orientation.** The main effort is given to deliver a quality product that brings value to the customer, and to avoid contention with customer.
- **Communication emphasis.** The emphasis is given to an informal, open communication of all participants in the project (the team and the customer).
- **Simplicity, informality.** An effort to simplify the process, make it efficient and get rid of all superfluous costs and activities is apparent in all the methodologies.
- **Incremental development with short iterations.** Development is always performed in small functional parts and the completed system (or its prototype) is delivered to the customer as soon as possible.
- **Modern technologies utilization.** Although not being a crucial constraint, AM utilize modern object-oriented programming languages and modern development environment possibilities.

Every single methodology of both groups brings useful practices that can be successfully utilized under specific circumstances. However, two negative situations may occur, when sticking to one methodology:

1. not utilizing useful practices of other methodologies,
2. using prescribed practices that do not bring advantage to the project (P o p p e n d i e c k , P o p p e n d i e c k , 2003).

Combining methodologies is a good solution, but it must be always done based on real needs. The presented method introduces a way how to evaluate the need of adaptation in various parts of a project. This adaptation can be most easily performed by adopting the practices of the particular methodologies that address the related subject.

CONCLUSION

In this paper, we addressed the issue that the New Economy and other aspects of today's dynamic world bring high demands on information systems in agriculture. There are many methodologies that differ in various aspects, mostly in the ratio of rigour and control to flexibility. Every project in a certain environment has a unique ideal set-up. Approximation to this set-up is a vital factor in a project's success. It means there must be some sort of software project structure adaptation. The first step is qualification and quantification of those adaptation needs.

The presented method brings one possible approach to this issue. It is based on the analogy between the system modelling and a software project and sees the software project as a system with inputs, outputs and inner structure, which is represented by project factors crucial in the project management.

The presented example shows how the method may be used for adaptation needs evaluations. In this simple example, the conclusions may be of course made just with common sense, but in real situations, typically tens of inputs and factors will be involved and the conclusions will not be that apparent.

The most important step is the selection of appropriate inputs and project factors sets. There should be the largest manageable number of independent elements in each set.

Using the method does not solve the software engineering issues, but it does help to work with them in a formal, structured, manageable way that simplifies discussion and reasoning and makes decision making process a documentable, traceable action. This brings an opportunity of costs reduction, because incorrect management decisions are the most expensive ones.

REFERENCES

KELLY, K.: New Rules for the New Economy. Penguin (Non-Classics) 1999.

POPPENDIECK, M. – POPPENDIECK, T.: Lean Software Development: An Agile Toolkit for Software Development Managers. Addison-Wesley 2003.

SKYTTNER, L.: General Systems Theory. World Scientific Publishing Company, 2001.

STANDISH Group International: Statistics about software projects failure rate. Available at: http://www.standishgroup..com/sample_research/PDFpages/chaos1994.pdf.

Received for publication on January 14, 2008
Accepted for publication on March 3, 2008

PERGL, R. (Česká zemědělská univerzita, Fakulta provozně ekonomická, katedra informačního inženýrství, Praha, Česká republika):

Řízení softwarových projektů v prostředí Nové ekonomiky.

Scientia Agric. Bohem., *39*, 2008: 55–60.

Zemědělská výroba v prostředí Nové ekonomiky vyžaduje kvalitní technologickou a informační podporu. Spolu s tímto rostou nároky na bezpečnost, dostupnost, ovladatelnost a funkčnost informačních systémů, což klade zvýšené nároky na řízení softwarových projektů. Vysoká míra neúspěšnosti ukazuje, že situace není zcela ideální.

Problémem není nedostatek metodologie, právě naopak, metodik pro řízení softwarových projektů existuje celá řada, a to jak tradičních, tak nových. Metodiky dělíme obecně na tzv. rigorózní, jež kladou důraz na procesy a řízení, a tzv. agilní, jež se snaží dosáhnout flexibility. Ideální struktura projektu a jeho řízení by měla být taková, aby výstupem projektu byl softwarový produkt v požadované jakosti při nepřekročení plánovaných zdrojů.

Příspěvek představuje jednu z možných cest, jak určit, které oblasti projektového řízení je třeba upravit. Je zde představen teoretický aparát, jež je založen na analogii systémového modelování a softwarového projektu. Na softwarový projekt nahlíží jako na systém, jež má vstupy, výstupy a vnitřní strukturu. Cílem je v souladu s teorií o komplexních adaptivních systémech provést adaptaci vnitřní struktury tak, aby výstupy (tj. produkt) odpovídaly vstupům (tj. zadání). V příspěvku je popsán aparát a je demonstrován na jednoduchém praktickém příkladu. Vstupem aparátu jsou vstupní požadavky na informační systém, jež kladou určité nároky na části projektu. Tyto nároky se sčítají, mohou však být zmírněny substitučními vazbami uvnitř projektu. Výstupem je kvantifikace částí projektu, jež na základě vstupů budou vyžadovat adaptaci.

Použití aparátu neřeší problémy řízení softwarového projektu, ale umožňuje s nimi formálním, strukturovaným a řízeným způsobem pracovat. Zjednodušuje diskusi a odvozování závěrů a přispívá k vytváření kvalifikovaných rozhodnutí založených na kvantifikovaných veličinách. Takto přispívá k eliminaci nesprávných rozhodnutí, a tím i k úspoře nákladů.

softwarové inženýrství; vývoj zemědělských informačních systémů; řízení softwarových projektů; systémové modelování; komplexní adaptivní systémy

Contact Address:

Ing. Robert P e r g l, Česká zemědělská univerzita v Praze, Fakulta provozně ekonomická, katedra informačního inženýrství, Kamýcká 1076, 165 21 Praha 6-Suchdol, Česká republika, tel.: +420 224 383 244, e-mail: pergl@pef.czu.cz