



INFORMATION TECHNOLOGY
RESEARCH INSTITUTE

**VERDE – A CHECKLIST-BASED
INFORMATION SYSTEM CHANGE
DECISION MAKING SUPPORT**
ELTIS-project

Version: 1.5	Classification: Public
Title: VERDE – A Checklist-based Information System Change Decision Making Support	Date: 2006-01-02 Status: Complete
Authors: Jussi Koskinen, Jarmo Ahonen, Irja Kankaanpää, Heikki Lintinen, Henna Sivula, Tero Tilus	

VERSION HISTORY

Version	Date	Modifications since last version
1.0	2004-10-25	Accepted by the project steering committee. Comments from the steering committee have been taken into account.
1.1	2004-11-11	Changes considering formulations and terminology which came up during inspection.
1.2	2004-11-12	Reorganization of criteria in Appendix 2 (p. 26)
1.3	2004-11-18	Minor changes to criteria appendices
1.4	2004-11-23	Removed redundancies in criteria appendices
1.5	2006-01-02	Updated headings to reflect method name

CONTENTS

1	Introduction.....	1
2	Use of VERDE.....	2
2.1	<i>Users</i>	2
2.2	<i>Use cases.....</i>	2
2.3	<i>Binding VERDE to IS change process.....</i>	4
2.4	<i>Self-direction</i>	5
3	Checklist	6
4	Checklist description	10
4.1	<i>Identification of the background factors</i>	10
4.1.1	<i>The need and motivation for IS change</i>	10
4.2	<i>Evaluation of the current system</i>	11
4.2.1	<i>Business value</i>	11
4.2.2	<i>Technical value.....</i>	11
4.3	<i>Objectives.....</i>	12
4.3.1	<i>Requirements.....</i>	13
4.3.2	<i>Strategy.....</i>	14
4.4	<i>IS change.....</i>	14
4.4.1	<i>IS change options</i>	14
4.4.2	<i>IS change evaluation.....</i>	16
4.5	<i>Evaluation of the target system</i>	16
4.5.1	<i>Target system – evaluating business value</i>	16
4.5.2	<i>Target system – evaluating technical value</i>	16
4.6	<i>Argumentation.....</i>	17
4.6.1	<i>Criteria.....</i>	17
4.6.2	<i>Arguments</i>	18
4.6.3	<i>Viewpoints.....</i>	19
4.6.4	<i>Misleading argumentation</i>	19
4.7	<i>Recording and follow-up.....</i>	20
4.7.1	<i>Evaluation of the IS change outcome</i>	21
4.7.2	<i>Follow-up</i>	21
5	Summary	22
	References	23
	Appendices	25
	<i>Appendix 1. Risks and success factors.....</i>	25
	<i>Appendix 2. System evaluation criteria</i>	26
	<i>Appendix 3. IS change evaluation criteria.....</i>	29

1 INTRODUCTION

This paper represents VERDE (*Verifiable Decision*) – a checklist-based method for IS change decision making support developed during ELTIS (*Extending the LifeTime of Information Systems*) -project. The project is funded by companies of Finnish software industry. Our choice of developing a checklist method was due to the need of having a simple-to-use way to evaluate a modernization case during the initial phase of making a decision. During the project we have earlier conducted an interview-based study of modernization decision making (Lintinen *et al.*, 2004). During this earlier study 29 persons were interviewed of which most were upper or middle-level managers responsible of software modernization decisions. Only few of them were completely satisfied with present support. A checklist of the top issues to be considered was mentioned most often as a possible support method. Another main observation was that expert judgment plays a significant role in decision making process.

The method presented here is needed for multiple purposes. The use of the method will be illustrated with two use cases. Users of the method are decision makers at any level of software change process. The structure of the method is hierarchical on three levels. Essentially the method consists of answering to questions, which are simple in their form and characterize the information system change situation. The content of the checklist is based on observations collected from the literature. The form of the checklist is based on the results of iterative internal reviews within the ELTIS-project. The method consists of the following primary parts: identification of the background factors (of the change), objectives, argumentation, and recording/follow-up. The form and content will later be reviewed within the participating software companies and feedback is to be gathered and the content and structure of the method revised accordingly while necessary. We have also developed and will further develop other methods for evaluating modernization situations, including MODEST (Koskinen *et al.*, 2004).

The paper is organized as follows: Section 2 illustrates the use of the method, Section 3 provides a listing of the questions to be used while applying the method, Section 4 describes the method in detailed level and provides links to the relevant theoretical background. Finally, Section 5 provides a summary. Appendices provide further information of other relevant methods, and detailed system evaluation criteria and IS change evaluation criteria.

2 USE OF VERDE

The purpose of VERDE is to

- aid decision making process concerning IS change activities
- promote successful IS change
- provide an extensive issue list of factors concerning IS change decisions
- increase effective supervision of the decision making process

Checklist provides neutral arguments and viewpoints, which help to examine decision making tasks from different perspectives. It serves decision making by providing a systematic and flexible way to reflect the argumentation and other steps of decision making. The structure of the checklist is designed in a way that it can be used as a comprehensive tool in overall modernization planning or as a complementary tool to support decision making only when needed.

Checklist consists of three (3) levels. The main level (the 1st level) can be used as a higher level issue list to confirm that the relevant issues concerning the IS change case at hand have been evaluated. This may take place, for instance, before IS change project is initiated and is usually done by middle-level manager or project manager (depending on the structure of the organization). The 2nd level specifies the main questions presented at the 1st level. These sub-questions are designed to be used to promote decision making in corresponding phases of maintenance process. Questions on the 3rd level are designed to help the user to answer to the second level questions and promote deeper reasoning of the issue.

Nevertheless, there are no rules how one should use the checklist. It is ultimately up to the user how to apply it in the best possible way. It should be noted also, that checklist should be adjusted to fit organization's processes and needs before use. It should not be applied blindfold. It is a tool for overall decision making and should be treated accordingly. It can help to produce correct decisions but is cannot create them independently.

2.1 Users

The primary user group of the checklist is decision-makers at any level of software change process. A checklist user can be, for instance, project manager, technical architect, IT manager, product manager or system designer. Additionally, checklist is a tool for preparative work before actual decision making.

2.2 Use cases

Possible use situations of the checklist throughout the change process have been illustrated with the following cases (FIGURE 1 and FIGURE 2). The phases of change process vary greatly from one organization to another, and therefore the examples below cannot be necessarily applied as such.

Name: Evaluating the need for software change

Actor(s): IT manager, product manager

Short description: Identification and evaluation of the background factors and motivators

Triggered by: Pressure for change; can be either external or internal

Actions: See checklist, section 4.1. Select the best option from the following actions (1-3):

1. If the actor is in a hurry or very confident about the need for the change, then he/she skims through the first level question. The issue and question are:

Has it been ensured that all interest groups share a common understanding of the system's capabilities and operation?

Is there a true motivation for the IS change?

2. If the actor prefers guidance in decision making, he/she should think through the issues on the first level and then answer the questions on the second level. The issues and questions are:

Has it been ensured that all interest groups share a common understanding of the system's capabilities and operation?

Is there a true motivation for the IS change?

Is the motivation arguable?

Was the change request handled according to the procedures agreed within the organization?

3. If the actor feels insecure about the need for system change, or is making the decision for the first time, then it is recommended to think through the issues on the first level and then answer the questions on the second level by using the questions on the third level as guidance. The issues and questions are:

Has it been ensured that all interest groups share a common understanding of the system's capabilities and operation?

Is there a true motivation for the IS change?

Is the motivation arguable?

What are the need and motivation for the change?

Is the motivation unambiguous to all stakeholders?

What follows if the need is not met?

Was the change request handled according to the procedures agreed within the organization?

If no, does it have any effect on the handling of the issue later on?

Where are the origins of the IS change? How and by whom it was initiated?

How the need was recognized? Could we have failed to recognize it?

As a result from each option (1-3), the actor has considered relevant topics concerning the change and has formed a conception of the need for software change. This conception can be used to support related decisions.

FIGURE 1. Example use case: Evaluating the need for software change

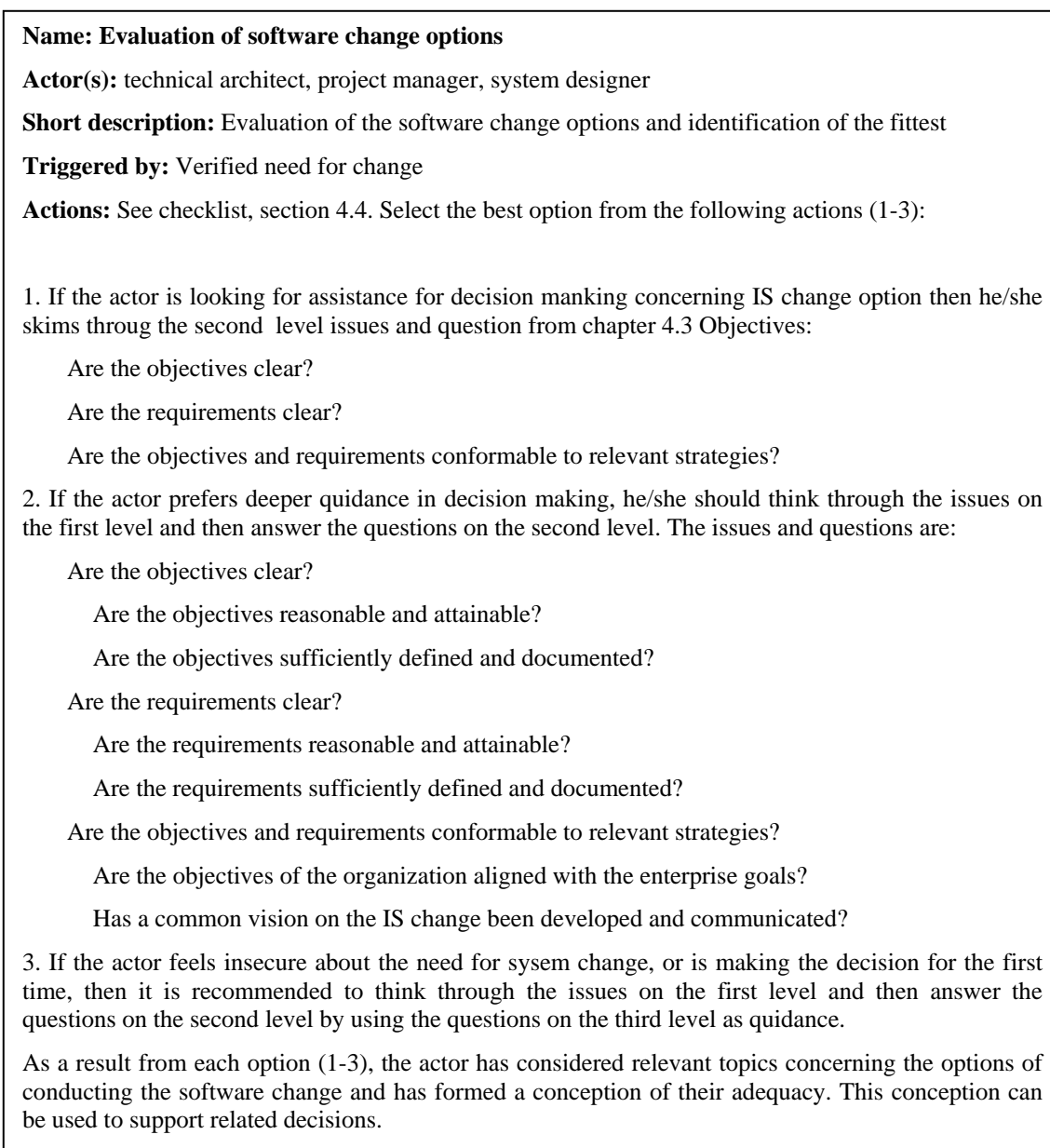


FIGURE 2. Example use case: Evaluation of software change options

2.3 Binding VERDE to IS change process

The question of what specific questions should be asked at what point of IS change process is quite naturally not answered by the checklist itself. The answer depends on the process and the organization in question and thus can not be given a “one size fits all” solution.

One possibility of systematic “implementation” of checklist is to use “implementation matrix” (see an example in FIGURE 3) where columns represent the phases or tasks of the process where checklist contents are to be bound to and rows represent checklist contents. Bindings are marked by checking the cells of the matrix. Coverage and sufficiently even distribution over checklist and process are easily observable from the matrix.

	Need recognition	Option analysis	Decision	Implementation	Evaluation
Background factors	X				
Current system	X				
Objectives		X			
IS change		X			
Target system		X			
Argumentation			X		
Recording and follow-up			X	X	X

FIGURE 3. Example of implementation matrix

2.4 Self-direction

We suggest that the use of the method should be self-directive (and feedback preserving). Use of the tool should elaborate itself for example by organizing lists, selecting and weighting criteria and questions and accumulating patterns and anti-patterns. Support for self-directiveness will not be implemented at this stage of development.

3 CHECKLIST

This chapter is a summary of the checklist questions in order of their appearance in chapter 4, where more thorough description of the background and motivation for the questions is given.

4.1 Identification of the background factors.....10

Has it been ensured that all interest groups share a common understanding of the system's capabilities and operation?

Is there a true motivation for the IS change?

Is the motivation arguable?

What are the need and motivation for the IS change?

Is the motivation unambiguous to all stakeholders?

What follows if the need is not met?

Was the IS change request handled according to the procedures agreed within the organization?

If no, does it have any effect on the handling of the issue later on?

Where are the origins of the IS change? How and by whom it was initiated?

How the need was recognized? Could we have failed to recognize it?

4.2 Evaluation of the current system11

Has the business value of the current system been evaluated?

Has the system been evaluated from the point of view of direct and indirect users?

Has the system been evaluated from the managers', end users' and customers' point of view?

Has the economic value of the system been evaluated?

Has the data value of the system been evaluated?

Has the system quality in use been evaluated?

Has the specialization value of the system been evaluated?

Has the technical value of the current system been evaluated?

Has the current system been evaluated from the point of view of technical users?

Has maintainability of the system been evaluated?

Has the degradation of the system been evaluated?

Has the obsolescence of the system been evaluated?

Has the reliability of the system been evaluated?

How stable is the system's operation? Have the unresolved problem reports and change requests been reviewed for trend information?

Have all user interfaces been identified?

Have critical algorithms been identified and analyzed?

Have the performance characteristics of the system been assessed?

Are the source code, library elements and build scripts available and current?

Have dependencies on undocumented features been identified?

Have the complexity and brittleness of the system been assessed?

Has the integrity of the current system been affected adversely by the maintenance?

Has the operational environment of the system been evaluated?

Are all customers and user groups identified?

Are all system products and services on which the users depend identified?

Can the system workload be accurately characterized?

Are all the external artifacts, system files and procedures on which the users depend identified?

Are there operational user scenarios to ensure that users understand system's operations and capabilities?

Is there a description of the sub-systems and their interfaces?

Are all external system interfaces identified and documented?

Are the hardware and software interoperability dependencies with external sites identified and documented?

Are software communication protocols identified and documented?

4.3 Objectives.....12

Are the objectives clear?

- Are the objectives reasonable and attainable?
 - What are the objectives?
 - Are there defined criteria for the successful accomplishment of objectives? Are these criteria measurable?
 - In what time period the objectives are attained?
- Are the objectives sufficiently defined and documented?
 - Is there an established procedure for performing business/mission needs elicitation to determine how new customer needs can best be met?
 - Are the objectives prioritized?

Are the requirements clear?

- Are the requirements reasonable and attainable?
 - What are the requirements?
- Are the requirements sufficiently defined and documented?
 - Is each requirement consistent with the objectives?
 - Are the requirements traceable to overall system objectives?
 - Is each requirement testable, once implemented?
 - What are the metrics to verify the achievement of requirements?
 - Have all requirements been specified at a proper level of abstraction?
 - Are the requirement prioritized?
 - How are the customer and user requirements prioritized?
 - Is the requirement really necessary, or does it represent an addition on feature that may not be essential to the objective of the system?
 - Is each requirement bounded and unambiguous?
 - Is the source (e.g. a person, a regulation, a document) of a requirement identified?
 - What other requirements relate to this requirement?
 - Do any requirements conflict with other requirements?
 - Does the requirement violate any application domain constraints?

Are the objectives and requirements conformable to relevant strategies?

- What are the relevant strategies at organization, corporate and enterprise levels?
- What is the corporate information technology strategy?
- Has a common vision on the IS change been developed and communicated?
 - Have the key decision makers and stakeholders been identified?
 - What is the overall scope of the system evolution effort?

4.4 IS change..... 14

Has sufficient amount of options (to pursue the objectives) been discovered and investigated?

- Have the details of selected option been recorded?
 - What methods were used to discover options?
 - Have risks and difficulties related to each option been analyzed?
- Is technology transition (if considered) justifiable?
 - Is the new technology sufficiently mature and stable?
 - Have pilot efforts or case studies confirmed the suitability of the technology for the specific application domain?
 - What is the potential impact of not adopting the technology?
 - Does the new technology have the potential to make a significant contribution to the enterprise goals and objectives?
 - Is the new technology a prerequisite for the system evolution effort?
 - Is the new technology required for system compatibility?
 - Is the new technology a prerequisite for adopting other technologies?
 - Have the benefits of the candidate technology been quantified?

Does the organization have sufficient resources and preparedness for the IS change?

- Have the risks been detected and mitigated?
- Are all interest groups committed to IS change?
- Has a common concept of operations for the proposed system been developed and communicated?

Have the benefits of IS change been assessed?

4.5 Evaluation of the target system..... 16

Has the business value of the target system been evaluated?

Has the target system been evaluated from the point of view of direct and indirect users?

See questions in section 4.2.1

Has the technical value of the target system been evaluated?

Has the target system been evaluated from the point of view of technical users?

What estimation models are used in evaluation?

Have operational scenarios been developed to describe how the proposed system will operate?

Are there standards with which the target system must comply?

Should the system be re-hosted on a new platform or operating system? Is the use of a new programming language justified?

Has maintainability of the target system been evaluated?

Has the degradation of the target system been evaluated?

Has the obsolescence of the target system been evaluated?

Has the reliability of the target system been evaluated?

Has the target operational environment been evaluated?

What changes are required in the operational environment to accommodate the new target system requirements?

What is the project impact of the proposed changes on the current business operations? How will these affect the customer and the organization?

What differences are there between the current system environment and the proposed target environment?

Do the customer and user requirements include explicit changes to the operational environment?

What is the projected impact of the proposed changes on performance and availability?

Which external interfaces need to be modified?

What testing is needed to assure interoperability?

What is the plan for “roll out” and “cut-over” to the new system?

What parts of the target and current systems need to coexist during operational transition?

In the event of crisis, to what degree can support be rolled back to the current operational environment?

Will the new target environment impose new operating procedures?

Will operators or system administrators require training on the new operating environment?

Have training needs been identified for customers and users of the system?

What changes are to be expected to the operational environment in the future? Will the changes impose new requirements to the target system?

4.6 Argumentation 17

Are all important aspects included in argumentation?

Are all relevant criteria included in argumentation?

What are the decision criteria in this particular case? Why?

What is the general framework according to which the decision is to be made?

Has sufficient attention been paid to critical success factors?

Are presented arguments valid?

What assumptions have been made concerning the change?

Are the assumptions reasonable and arguable?

Are expectations outlined and understood ?

Are drivers and priorities identified and understood ?

Are presented arguments measurable?

Are the costs, pros and cons of the IS change pursued evenly?

Has the possibility of “harmful” arguments taken into account?

Has the possible side-effects of the arguments been detected and evaluated?

Are all viewpoints of all interest groups taken into account?

Have all interest groups been able (and given a chance) to state and defend their viewpoint?

Is partiality of all interest groups sufficiently noted?

How the partiality is handled / mitigated?

Have sufficient attempts made to avoid argumentative illusions?

Is there an illusion created by appealing to reciprocation?

Is there an illusion created by appealing to consistency?

Is there an illusion created by appealing to social validation?

Is there an illusion created by appealing to liking?

- Is there an illusion created by appealing to authority?
- Is there an illusion created by appealing to scarcity?
- Is there an illusion of an overly spacious time-table (i.e. the implementation of little changes and additions can easily fit in to the original software change plan)?
- Is there an illusion of the simplicity of the yet non-existent?
- Is there an illusion that an on-going project cannot be put to an end, even if the apparent outcome would be disastrous, because the resources spent would be wasted?

4.7 Recording and follow-up20

Is it possible to learn from this decision?

- Are the decision and context properly documented?
- Is it possible to evaluate the decision?
 - Who is responsible of evaluation?
 - Is it possible to evaluate the correctness of the decision?
 - What is planned to happen if evaluation shows something remarkable with the outcome?
 - What is the validation criteria for IS change success?
- Is it possible to do a proper follow-up and learn from the gathered information?
 - Who is responsible of follow-up?
 - Will it be possible to compare this and previous IS change efforts (and decisions)?

4 CHECKLIST DESCRIPTION

The structure of this tool is hierarchical. The first level of hierarchy, formulated as 1-3 questions per entity, constitute a single checklist. In deeper levels, it expands to several more narrow scoped and fine grained checklists. The collection of all the questions in the order of appearance can be found from chapter 3.

The questions on levels 1 to 3 are presented with the following notation:

1st level questions

2nd level questions

3rd level questions

IS change risks and success factors are presented in Appendix 3.

4.1 Identification of the background factors

Has it been ensured that all interest groups share a common understanding of the system's capabilities and operation?

Bergey et al. (1997) present a framework for disciplined legacy system evolution in which the evaluation of system factors (i.e. the core system, system operational environment and support environments) is an initial step for successful change process. Before change can take place it is necessary to understand the current state of the system.

4.1.1 The need and motivation for IS change

Is there a true motivation for the IS change?

Is the motivation arguable?

What are the need and motivation for the IS change?

Is the motivation unambiguous to all stakeholders?

What follows if the need is not met?

Lehman *et al.* (1998) have formulated and represented based on empirical data the so-called Lehman's laws. According to Lehman's first law software must be continually adapted or it will become progressively less satisfactory in "real-world" environments.

Reengineering project "radicalism" and degree of changes in the work-flow patterns correlate positively with project success (Teng *et al.*, 1998). This is due to the fact that often only major positive changes in business processes justify the costs of reengineering efforts.

Was the IS change request handled according to the procedures agreed within the organization?

There should be a well-defined reengineering process. Bergey *et al.* (1999) identify as an antipattern the situation where the consideration of the potential problems is simply passed by the slogan: “We have our best people working on it”. Even the definition of the process is not enough. Pressman (1992) states that the adequacy of having standard-procedures gathered into the organization’s quality assurance handbook for information system development is a myth pestering especially management. Completeness, adequacy, timeliness and level of following those procedures typically are limited.

If no, does it have any effect on the handling of the issue later on?

Frequent bypassing of the agreed procedures could be a consequence of not knowing about them or not agreeing on the necessity of them. In that case it might be advisable to evaluate the reason for the bypass and the need to mitigate the possible problems.

Where are the origins of the IS change? How and by whom it was initiated?

How the need was recognized? Could we have failed to recognize it?

4.2 Evaluation of the current system

4.2.1 Business value

Has the business value of the current system been evaluated?

Has the system been evaluated from the point of view of direct and indirect users?

Has the system been evaluated from the managers’, end users’ and customers’ point of view? Has the economic value of the system been evaluated?

Has the data value of the system been evaluated?

Has the system quality in use been evaluated?

Has the specialization value of the system been evaluated?

4.2.2 Technical value

Has the technical value of the current system been evaluated?

Has the current system been evaluated from the point of view of technical users?

Has maintainability of the system been evaluated?

Has the degradation of the system been evaluated?

Has the obsolescence of the system been evaluated?

Has the reliability of the system been evaluated?

How stable is the system’s operation? Have the unresolved problem reports and change requests been reviewed for trend information?

Have all user interfaces been identified?

Have critical algorithms been identified and analyzed?

Have the performance characteristics of the system been assessed?

Are the source code, library elements and build scripts available and current?

Have dependencies on undocumented features been identified?

Have the complexity and brittleness of the system been assessed?

Has the integrity of the current system been affected adversely by the maintenance?

Has the operational environment of the system been evaluated?

Are all customers and user groups identified?

Are all system products and services on which the users depend identified?

Can the system workload be accurately characterized?

Are all the external artifacts, system files and procedures on which the users depend identified?

Are there operational user scenarios to ensure that users understand system's operations and capabilities?

Is there a description of the sub-systems and their interfaces?

Are all external system interfaces identified and documented?

Are the hardware and software interoperability dependencies with external sites identified and documented?

Are software communication protocols identified and documented?

[FIXME: Oman & Hagemester also include management (of software supplier) as an influencing factor of software maintainability. We exclude it here due the fact that the information is either "external constant" (one supplier considered) or not available (several suppliers → information is not provided).]

Evaluation criteria is included in Appendix 2.

4.3 Objectives

Are the objectives clear?

Are the objectives reasonable and attainable?

What are the objectives?

Are there defined criteria for the successful accomplishment of objectives? Are these criteria measurable?

In what time period the objectives are attained?

Are the objectives sufficiently defined and documented?

Is there an established procedure for performing business/mission needs elicitation to determine how new customer needs can best be met? Are the objectives prioritized?

4.3.1 Requirements

Are the requirements clear?

Bergey *et al.* (1999) identify poorly refined requirements as an anti-pattern, characterized by slogan: “Our needs are simple”. System requirements should be elicited and validated properly. Also Pressman (1992) identifies the following antipattern which is often pestering IS customers: “General specification of goals is sufficient for starting programming, specifications can be detailed later”. In reality poor requirements definition is a very common reason for all IS project failures. Customers also tend to overestimate the flexibility of making changes into software. It is true that software is much more flexible than hardware, but project costs are highly elevated if changes need to be made in implementation phase as compared to the situation where changes are made already at the requirements specification or system design phases.

Are the requirements reasonable and attainable?

What are the requirements?

Are the requirements sufficiently defined and documented?

Is each requirement consistent with the objectives?

Are the requirements traceable to overall system objectives?

Is each requirement testable, once implemented?

What are the metrics to verify the achievement of requirements?

Have all requirements been specified at a proper level of abstraction?

Are the requirement prioritized?

How are the customer and user requirements prioritized?

Is the requirement really necessary, or does it represent an addition on feature that may not be essential to the objective of the system?

Is each requirement bounded and unambiguous?

Is the source (e.g. a person, a regulation, a document) of a requirement identified?

What other requirements relate to this requirement?

Do any requirements conflict with other requirements?

Does the requirement violate any application domain constraints?

Weinberg (1982; 1992) have noted the central importance of profoundly understanding the problem to be solved by information technology. One of the main problems may be the lack of sufficient feedback in early stages of new system development or major evolution. Customers typically don't know exactly what they want before having feedback in form of a usable implementation. On the other hand the cost of large implementations is very high. Thus in case of very large scale changes prototyping may be a lucrative option to gather the needed feedback quickly (Davis, 1995).

One of the related problems is need for quality assurance. Failure of meeting set quality goals due to their poor definition is identified by Sneed (1999) as an important risk

factor in reengineering projects. One of the myths of programmers is the assertion that system quality can not be reliably evaluated before the actual program has been implemented (Pressman, 1992). This is partially true, but especially in case of very high quality requirements, also the use of code and document inspections and reviews as well as use of formal methods should be considered.

4.3.2 Strategy

Are the objectives and requirements conformable to relevant strategies?

What are the relevant strategies at organization, corporate and enterprise levels?

What is the corporate information technology strategy?

Has a common vision on the IS change been developed and communicated?

Have the key decision makers and stakeholders been identified?

What is the overall scope of the system evolution effort?

Bergey *et al.* (1999) give management's lack of long-term commitment in planning as an example of antipattern as a slogan: "Tomorrow is another day". This is especially problematic when managers are in quick rotation, in which case long-term plans can not be truly relied on.

Bergey *et al.* (1997) suggest using an enterprise framework for characterizing the system evolution environment because it takes into account strategic and business operations, and lines up the organizational, engineering, technology and systems elements with them.

4.4 IS change

4.4.1 IS change options

Has sufficient amount of options (to pursue the objectives) been discovered and investigated?

Have the details of selected option been recorded?

What methods were used to discover options?

Have risks and difficulties related to each option been analyzed?

Bergey *et al.* (2001) have present Options Analysis for Reengineering (OAR) method to support component re-use and mining decisions. As a part of it, selecting mining options includes five central tasks: 1) Determinate drivers for selection of each option, and select the best option(s), 2) Verify selected option by recording all the details of it, 3) Compare selected options to the required characteristics of potential legacy components, 4) Present findings, and 5) Prepare a report of the decision and the rationales for the selected option(s). (Bergey *et al.*, 2001)

Bergey *et al.* (1999) mention the slogan of: "We're too busy to plan" as an antipattern. There should be adequate plans and also necessary resolve to follow the plans. Large

scale modernizations should also be planned and designed according to the acknowledged virtues of good actual system development projects. Parnas & Clements (1986) have emphasized the importance of design. Especially they have emphasized the importance of faking the idealized design process and production of appropriate documentation because of the benefits for communication, management and review. These considerations should be taken into account also in case of large-scale modernization projects. They also characterize the inherent difficulty of system design: 1) even though that system requirements would be precisely defined during the design of the needed changes of the implementation, in reality there will be changes which cause changes also to the plans, 2) even though that all factors affecting system design and implementation would be known, their amount and complexity can not be handled error-freely, 3) even if all details could be managed, non-trivial projects include changes and human errors, 4) in principle reuse is very favorable, but in reality reuse of design patterns or system components may cause contradictions with the project requirements. Thus, due to these inherent problems too optimistic stances towards the easiness of modernization should be avoided.

Is technology transition (if considered) justifiable?

Is the new technology sufficiently mature and stable?

Have pilot efforts or case studies confirmed the suitability of the technology for the specific application domain?

What is the potential impact of not adopting the technology?

Does the new technology have the potential to make a significant contribution to the enterprise goals and objectives?

Is the new technology a prerequisite for the system evolution effort?

Is the new technology required for system compatibility?

Is the new technology a prerequisite for adopting other technologies?

Have the benefits of the candidate technology been quantified?

Bergey *et al.* (1999) identifies the situation where management predetermines technical decisions as an antipattern.

Weinberg (1982; 1992) presents the following advise based on extensive software engineering experience: 1) those who do not rely too much on latest technological gimmicks but are still ready to try new ideas will succeed best, 2) no solution is best to every situation, 3) solution best suited to a particular situation may be the worst in another situation, 4) there are many useful approaches which work in many situations, 5) it is useful to get to know such approaches which have worked well in the past, 6) the crux of solution is not only on what approaches are applied (know-how) but rather when they are to be applied (know-when) thus enabling adapting the approach to the peculiarities of the problem to be solved.

Examples of modernization methods and modernization support technologies include: reverse engineering, reengineering, re-factoring, re-documentation, systematic configuration management and automated migration (Koskinen *et al.*, 2004b). However, Pressman (1992) identifies as a typical management antipattern the overemphasis of the importance of system development tools. Good development tools are important, but as

such their installation and even use is not at all a sufficient condition for successful system development or modernization. Similarly Pressman (1992) also notes another typical management antipattern of relying on adding programmers to a delayed project. According to the so-called Brook's Law, this in fact delays the project further due to the necessary overhead of consultation and communication within the project group.

4.4.2 IS change evaluation

Does the organization have sufficient resources and preparedness for the IS change?

Have the risks been detected and mitigated?

Are all interest groups committed to IS change?

Has a common concept of operations for the proposed system been developed and communicated?

Have the benefits of IS change been assessed?

IS change evaluation criteria is included in Appendix 3.

4.5 Evaluation of the target system

4.5.1 Target system – evaluating business value

Has the business value of the target system been evaluated?

Has the target system been evaluated from the point of view of direct and indirect users?

See questions in section 4.2.1

4.5.2 Target system – evaluating technical value

Has the technical value of the target system been evaluated?

Has the target system been evaluated from the point of view of technical users?

What estimation models are used in evaluation?

Have operational scenarios been developed to describe how the proposed system will operate?

Are there standards with which the target system must comply?

Should the system be re-hosted on a new platform or operating system? Is the use of a new programming language justified?

Has maintainability of the target system been evaluated? Has the degradation of the target system been evaluated? Has the obsolescence of the target system been evaluated? Has the reliability of the target system been evaluated?

Has the target operational environment been evaluated?

What changes are required in the operational environment to accommodate the new target system requirements?

What is the project impact of the proposed changes on the current business operations? How will these affect the customer and the organization?

What differences are there between the current system environment and the proposed target environment?

Do the customer and user requirements include explicit changes to the operational environment?

What is the projected impact of the proposed changes on performance and availability?

Which external interfaces need to be modified?

What testing is needed to assure interoperability?

What is the plan for “roll out” and “cut-over” to the new system?

What parts of the target and current systems need to coexist during operational transition?

In the event of crisis, to what degree can support be rolled back to the current operational environment?

Will the new target environment impose new operating procedures?

Will operators or system administrators require training on the new operating environment?

Have training needs been identified for customers and users of the system?

What changes are to be expected to the operational environment in the future? Will the changes impose new requirements to the target system?

Evaluation criteria is included in Appendix 2.

4.6 Argumentation

Are all important aspects included in argumentation?

Important aspects include: legislative regulation, software business processes, technologies, application area characteristics, user requirements, available human resources, applied process models, available software metrics and available system documentation (Koskinen *et al.*, 2004a).

4.6.1 Criteria

Are all relevant criteria included in argumentation?

What are the decision criteria in this particular case? Why?

What is the general framework according to which the decision is to be made?

Has sufficient attention been paid to critical success factors?

General frameworks or approaches which may be used in judging criteria include, TCO, ROI (Return on Investment) (Erdogmus, 2004), and reengineering planning process (Sneed, 1995).

Pressman (1992) identifies as a programmer antipattern the generally faulty assumption of the distribution of the needed effort in different system development phases, such that the delivery of the code to the customer would mark the end of needed work. In reality generally 50-75% of the total effort is required in maintenance phase (Sommerville, 1996). Thus, constant system evolution is the normal case while dealing with information systems with long-term success and long lifetime. Similarly Pressman (1992) notes the programmer antipattern of overemphasizing source code as a deliverable, while in reality the whole configuration including documentation is important. Faulty assumptions in these regards may lead to insufficient allocation of resources to a project and thus problems with budget and schedule.

4.6.2 Arguments

Are presented arguments valid?

What assumptions have been made concerning the change?

Are the assumptions reasonable and arguable?

Are expectations outlined and understood ?

Are drivers and priorities identified and understood ?

Bergey *et al.* (1999) give the following anti-pattern characterized by the claim: “We’ve a ‘bulletproof’ strategy”, while in reality real validation often is very difficult. The selected strategy may be flawed or incomplete.

Are presented arguments measurable?

Sneed (1995) lists numerous useful measures relevant in reengineering projects including: system size measures, system complexity metrics, system quality attributes, maintenance and operation costs, business value and user satisfaction. Ransom (2002) also mention that in order to do effective evolution planning one should calculate measures for system’s technical quality and business value.

Are the costs, pros and cons of the IS change pursued evenly?

Sneed (1995) proposes a 5-step process including: 1) portfolio justification, 2) portfolio analysis, 3) cost estimation, 4) cost-benefit analysis, and 5) contracting.

Sneed (1999) have for example identified following cases which correlate with elevated risk level in reengineering projects due to the lack of considering related costs adequately: 1) performance losses in migration (either between environments or programming languages), 2) time delays due to architectural mismatches, and 3) time delays due to testing bottlenecks.

Has the possibility of “harmful” arguments taken into account?

Has the possible side-effects of the arguments been detected and evaluated?

Care must be taken to ensure that arguments which will (eventually) get out are not likely to be (mis)interpreted in a way that would be harmful to the organization.

4.6.3 Viewpoints

Are all viewpoints of all interest groups taken into account?

Have all interest groups been able (and given a chance) to state and defend their viewpoint?

Is partiality of all interest groups sufficiently noted?

How the partiality is handled / mitigated?

In a case of legacy system decision making “it is essential to call upon many different roles (stakeholders) within the organization to frame a decision.” Decisions concerning system change are not plainly technical matters but require participation from different organizational levels and tasks. In SABA model, organizational scenarios and technology scenarios are used to assess decision making (Bennet et al., 1999)

Bergey *et al.* (1999) gives the following antipattern: “We’re known for our ‘on-the-job-training’”, as an example of not considering problems adequately. In this case the established level of training in the organization may not be sufficient for ensuring the success of a demanding reengineering project.

Bergey et al. (2001) note that stakeholder assumptions and constrains are major drivers influencing the mining reusable legacy system components.

4.6.4 Misleading argumentation

Have sufficient attempts made to avoid argumentative illusions?

Argumentative means of getting positive response are divided into six categories by the basic tendencies of human behavior they appeal to: reciprocation, consistency, social validation, liking, authority and scarcity. Cialdini (2004) describes these “rules of persuasion” and states that knowledge of them can truly be thought of as empowerment.

Is there an illusion created by appealing to reciprocation?

All societies subscribe to a norm that obligates individuals to repay in kind what they have received. When receiving a seemingly free offering (even if it was unsolicited or unwanted) we are convinced to somehow return the “favour”. The same mechanism works also the other way. Rejecting a request for help has the same effect than receiving a gift.

Is there an illusion created by appealing to consistency?

We have a natural tendency to behave consistently. For example signing a petition makes us more likely to actually promote that particular issue later on. Public commitment strengthens the effect of consistency.

Is there an illusion created by appealing to social validation?

One fundamental way to decide upon what to do is to parrot others. If many individuals have decided in favor of a particular idea, we are more likely to follow. We perceive that particular idea to be more valid.

Is there an illusion created by appealing to liking?

People prefer to say “yes” to those they like; those they have or would like to have a connection to. Compliments also stimulate liking, and even inaccurate praise may be effective.

Is there an illusion created by appealing to authority?

In social validation, the opinions of authorities are weighted above others. A person harnessing the power of authority by parading her experience or scientific credentials is not a problem when the claims are true. The problem comes when we are subjected to phony claims or make false assumptions based on authority symbols.

Is there an illusion created by appealing to scarcity?

A great deal of evidence shows that items and opportunities become more desirable to us as they become less available. One should be cautious when the desired option seems to be so elusive that one feels pressure to make a quick decision for it.

Is there an illusion of an overly spacious time-table (i.e. the implementation of little changes and additions can easily fit in to the original software change plan)?

Is there an illusion of the simplicity of the yet non-existent?

According to Pizka (2004) the complexity of the existing system and simplicity of the planned (yet non-existent) both tend to be overestimated. We have the tendency to be blind to problems not at hand, solved or not. Thus the required change may feel like a simple and easy task when it takes place far in the future or is not completely defined yet.

Is there an illusion that an on-going project cannot be put to an end, even if the apparent outcome would be disastrous, because the resources spent would be wasted?

Many of the examples and especially myths of software engineering noted by Pressman (1992) are part of (often unintentional) misleading argumentation based on faulty premises. Bergey *et al.* (1999) gives the following examples of this sort of anti-patterns in case of reengineering projects: 1) “We’re on top of it (*i.e.* the legacy system)”, which undoubtedly is a good objective and a nice slogan while in reality problems with maintenance may still be great even though system people do not necessarily want to emphasize them. 2) “Anybody can specify architecture”, while in reality it is a central and important consideration. 3) “We rely on experts to get us there”, while in reality some problems are too hard even to experts to solve (Weinberg, 1982; 1992).

4.7 Recording and follow-up

Is it possible to learn from this decision?

Are the decision and context properly documented?

Warren and Ransom (2002) suggest that when a method to support software system evolution, such as Renaissance, is used the decision criteria and possible change constraints should be recorded.

4.7.1 Evaluation of the IS change outcome

Is it possible to evaluate the decision?

Who is responsible of evaluation?

”Identifying responsibilities and assigning them to individuals is part of project planning” (Warren and Ransom, 2002).

Is it possible to evaluate the correctness of the decision?

What is planned to happen if evaluation shows something remarkable with the outcome?

In addition to the evaluation itself, the handling of the results of the evaluation should be well-defined too. Otherwise it is possible that uncomfortable issues are silently ignored, nothing is learned and the evaluation effort is wasted.

What is the validation criteria for IS change success?

At this point one should refer to the objectives 4.3 and argumentation 4.6 of the IS change at hand. They, when properly defined and recorded, should form the core of the validation criteria.

4.7.2 Follow-up

Is it possible to do a proper follow-up and learn from the gathered information?

Who is responsible of follow-up?

Will it be possible to compare this and previous IS change efforts (and decisions)?

Commensurability of the recorded data between this and other IS change processes is essential to the utility of the data in process improvement.

5 SUMMARY

We have represented a checklist-based method which supports decision making and early evaluation of an information system change initiative and provides extensive listings of relevant issues and points of view. The method has been developed based on the needs of the companies of Finnish software industry participating to ELTIS-project. We have detailed the use of the method, provided an overview of its hierarchical structure, detailed its content and provided links to the theoretical background. The model consists of a hierarchical question list. The method can be used at a selected level of granularity. Deeper level questions specify the nature of the situation. Appendices provide the most detailed-level information, including system evaluation criteria and IS change evaluation criteria. In future, we aim at testing the model in software industry, gathering feedback of its use, and revising it according to the received experiences.

REFERENCES

- Aversano, L., Esposito, R., Mallardo, T. & Tortorella, M. (2004). "Supporting decisions on the adoption of re-engineering technologies". *Proceedings of the Eighth European Conference on Software Maintenance and Reengineering (CSMR'2004)*, 95-104. IEEE Computer Soc.
- Bennett, K. H., Ramage, M. and Munro, M. (1999), "Decision model for legacy systems". *IEE Proceedings - Software* 146 (3), 153-159
- Bergey, J., Smith, D., Tilley, S., Weiderman, N. & Woods, S. (1999). "Why reengineering projects fail". Software Engineering Institute, Carnegie-Mellon University, Pittsburgh, PA, *CMU/SEI-99-TR-010*.
- Bergey, J. K., Northrop, L. M. and Smith, D. B. (1997). "Enterprise Framework for the Disciplined Evolution of Legacy Systems". Technical report CMU/SEI-97-TR-007, Carnegie-Mellon University, Pittsburgh, PA
- Bergey, J., O'Brien, L. and Smith, D. (2001). Options Analysis for Reengineering (OAR): A Method for Mining Legacy Assets. Software engineering institute. Report: CMU/SEI-99-TR-010, Carnegie-Mellon University, Pittsburgh, PA.
- Cialdini, R. B. (2004). "The Science of Persuasion". *Scientific American Mind* 14(1), 70-77.
- Davis, A. (1995). "Software prototyping". *Advances in Computers* 40, 39-63.
- Erdogmus, H., Favaro, J. & Strigel, W. (2004). "Return on Investment". *IEEE Software* 21 (3), 18-22.
- Koskinen, J., Ahonen, J., Tilus, T., Sivula, H., Lintinen, H. & Kekkonen, J. (2004). "MODEST: A Method for Early System Modernization Need Estimation". ELTIS-project, Information Technology Research Institute (ITRI), Univ. of Jyväskylä. Technical report. Available at: <URL: <http://titu.jyu.fi/eltis/papers/modest.pdf>>.
- Koskinen, J., Sivula, H., Lintinen, H. & Tilus, T. (2004a). "Evaluation of Software Evolution Options". *Publications of the Information Technology Research Institute* 14, University of Jyväskylä.
- Koskinen, J., Sivula, H., Lintinen, H. & Tilus, T. (2004b). "Evaluation of Software Modernization Estimation Methods Using NIMSAD Meta Framework". *Publications of the Information Technology Research Institute* 15, University of Jyväskylä.
- Lehman, M., Perry, D. & Ramil, J. (1998). "Implications of evolution metrics on software maintenance". *Proceedings of the International Conference on Software Maintenance - 1998*, 208-217. IEEE Computer Soc.
- Lintinen, H., Koskinen, J., Ahonen, J., Sivula, H. & Tilus, T. (2004). "Software Modernizations: A Qualitative Analysis of Industrial Decision Making". ELTIS-project, Information Technology Research Institute (ITRI), Univ. of Jyväskylä. Technical report.
- Parnas, D.L. & Clements, P.C. (1986). "A rational design process: How and why to fake it". *IEEE Transactions on Software Engineering* SE-12 (2), 251-257.
- Pitzka, M. (2004). "Adaptation of Large-Scale Open Source Software – An Experience Report". *Proceedings of the Eighth European Conference on Software Maintenance and Reengineering (CSMR'2004)*, 147-153. IEEE Computer Soc.
- Pressman, R. (1992). "Software Engineering: A Practitioner's Approach" (3rd ed.). McGraw-Hill.
- Sneed, H. (1995). "Planning the reengineering of legacy systems". *IEEE Software* 12 (1), 24-34.

- Sneed, H. (1999). "Risks involved in reengineering projects". *Proceedings of the IEEE Sixth Working Conference on Reverse Engineering*, 204-211. IEEE Computer Soc.
- Sommerville, I. (1996). "Software Engineering" (5th ed.). Addison-Wesley.
- Teng, T., Jeong, S. R. & Grover, V. (1998). "Profiling successful reengineering projects". *Communications of the ACM* **41** (6), 96-102.
- Warren, I.; Ransom, J. (2002), Renaissance: a method to support software system evolution. Proceedings. of the 26th Annual International Computer Software and Applications Conference, 2002(COMPSAC 2002), 415 – 420. IEEE Computer Society
- Weinberg, G. (1982). "Rethinking System Analysis & Design". Dorset House.
- Weinberg, G. (1992). "Quality Software Management, vol. 1: Systems Thinking". John Wiley & Sons.

APPENDICES

Appendix 1. Risks and success factors

Pressman (1992) gathers the following software engineering myths (*i.e.* faulty assumptions regarding the nature of software engineering, which have at least historically been often regarded as truths): 1) “Software quality handbook and standards suffice for system development”, 2) “Modern system development tools suffice”, 3) “In case of project delays the situation can be corrected by employing more work force to the project”, 4) “General specification of project objectives suffice, specification can be detailed later”, 5) “Needed changes due to the constantly changing project requirements can be easily implemented since software is flexible”, 6) When we (as programmers) have written the program and gotten it to run, our work is done”, 7) “It is impossible to evaluate quality before program runs”, 8) “The only deliverable of a successful project is a running program”.

Bergey *et al.* (1999) list the following anti-patterns in reengineering projects: 1) The organization inadvertently adopts a flawed or incomplete reengineering strategy (“We’ve a ‘bulletproof’ strategy”), 2) The organization makes inappropriate use of outside consultants and outside contractors (“We rely on experts to get us there”), 3) The work force is tied to old technologies with inadequate training programs (“We’re known for our ‘on-the-job-training’”), 4) The organization does not have its legacy system under control (“We’re on top of it (*i.e.* the legacy system)”), 5) There is too little elicitation and validation of requirements (“Our needs are simple”), 6) Software architecture is not a primary reengineering consideration (“Anybody can specify an architecture”), 7) There is no notion of a separate and distinct ‘reengineering process’ (in the organization) (“We have our best people working on it”), 8) There is inadequate planning or inadequate resolve to follow the plans (“We’re too busy to plan”), 9) Management lacks long-term commitment (“Tomorrow is another day”), 10) Management predetermines technical decisions (“If there’s one thing we’re good at, it’s giving orders”).

Sneed (1999) lists the following major risks in reengineering projects: 1) performance losses in migration (either between environments or programming languages), 2) time delays due to architectural mismatches, 3) time delays due to testing bottlenecks, 4) failure of meeting set quality goals due to their poor definition, and 5) resistance and dissatisfaction of user programmers

Warren and Ransom (2002) list the following risk factors: 1) lack of system knowledge, 2) lack of experienced maintenance personnel, 3) poor documentation, 4) new technology skills shortage, 5) legacy technology skills shortage, 6) errors introduced during evolution, 7) technology immaturity, 8) loss of embedded business rules, 9) system will not meet evolution requirements and 10) obsolete operational environment.

Appendix 2. System evaluation criteria

BUSINESS VALUE

ECONOMICAL VALUE

Operating cost

 Maintenance cost

 License cost

Redevelopment cost

Future utility

 Stability of the application area

 Stability of the business area

 Stability of the business operations

 Requirements volatility

 Expected lifetime of the system

 Life cycle of technology

 Prevalence of technology

 Portability

 Possibilities to future development

 Availability of skilled maintenance personnel

 Vendor support

Intensity of product use

DATA VALUE

Data criticality

Data dependence

Data quality

Data size

QUALITY IN USE

Functional adequacy

Stability

Accuracy

Interoperability

Usability

 Efficiency

 Availability

User satisfaction

Safety / security

SPECIALIZATION VALUE

Specialization level

 Type of software (administrative, embedded, real-time system, etc.)

 Generality

 Reusability

 Relationship with other application

External dependencies

TECHNICAL VALUE

MAINTAINABILITY

Age

Complexity

Programming language

Application

Size

Source code quality

Consistency

Structuredness

Modularity

Module cohesion

Module coupling

Nesting

Encapsulation

Accordance with design principles

Delocalization of the system logic

Traceability of design decisions

Amount of hard coding

Analyzability

Testability

Domain of the system

Number of variants

Number of versions

Documentation

Types of documentation (the following types of documentation should be examined from the point of view of presented quality attributes)

Source code comments

User manual

Other documentation

Descriptiveness

Accuracy

Consistency

Unambiguity

Completeness

Extent

Contents

Correctness

Traceability

Verifiability

Update level

Readability

Accessibility

Consistency

Typography

Comprehensibility

Structuredness

Clarity

- Compactness
- Modifiability
 - Partitions
 - Redundancies
 - Binding and distribution
 - Availability of electronic copy

Maintenance efficiency and quality

- Maintainers' experience
 - General maintenance experience
 - Technical work experience
 - Experience of application domain
 - Experience of the system

Maintainers' morale

- Maintenance difficulty
- Maintenance support techniques
- Complexity of maintenance requests
- Quality of support personnel
 - Skill levels of system support

Level of using established standards (e.g. programming languages, graphical interfaces, databases)

DEGRADATION

- Responsiveness degradation
- Reliability degradation
- Maintainability degradation
 - Adaptability degradation
 - Maintenance backlog
 - Availability of the original developers
 - Deterioration rate of code quality

OBSOLESCENCE

- Software obsolescence
 - Programming languages
 - Compilers
 - Debuggers
- Database obsolescence
- Operating system obsolescence
- Hardware/software infrastructure obsolescence [1, 2]

RELIABILITY

- Software reliability

Appendix 3. IS change evaluation criteria

IS CHANGE PROPERTIES

NEED

Necessity

Emergency

RELATIONSHIP WITH OTHER CHANGE REQUESTS

VALIDITY

Accordance with organizational ...
regulations,
strategies and
processes

Legality

INFLUENCE ON ...

processes

other systems

business partners

customers

SIZE OF THE CHANGE

TECHNOLOGICAL OPPORTUNITIES

Unification of platform/system portfolio

Increased ...

interoperability

compatibility

performance

maintainability

BENEFITS

Extended system lifetime

Reduced work time

Cost savings

Enhancement in customer service

Enhancement in data transmission

DISADVANTAGES

ORGANIZATIONAL ABILITIES

RESOURCES REQUIRED FOR IS CHANGE

Availability of skilled personnel

Team size

Programming language experience

General software development experience

Technical work experience
Experience of application domain
Experience of the system

Cost

Workload / effort
Influence to processes/other systems/business partners/customers
Repayment period
User training
General economic picture

Time

Tool support

ORGANIZATIONAL CHARACTERISTICS

Type of the organization
Users' skill level
Amalgamations and acquisitions
Imago
Possibility to collaborate with other organizations
Confidence to business partners
Customer's confidence to software vendor
Organizational attitude to change

TECHNICAL MATURITY OF THE ORGANIZATION

Programming practices
Software tools
Available software metrics
Training procedures
Configuration management
Structured analysis/design methods
Error-correction process model
Quality assurance techniques