The 7th International Scientific Conference
"DEFENSE RESOURCES MANAGEMENT IN THE 21st CENTURY"
Braşov, November 15th 2012

RISK MANAGEMENT IN MULTINATIONAL IT PROJECTS

Emil ŞCHIOPU

Romanian Ministry of National Defense

Abstract:
Cultural differences have become more perceivable at the same time as national borders have become less relevant and economic systems more dependent on each other. Current concepts about culture do not seem to help in understanding the differences and their effects in practice. Despite the previous efforts within the project management discipline, a large variety of concepts and the lack of practical solutions are leading to disregarding especially the innovation potential arising from multiculturalism.

Key words: risk, management, multinational, project, software, development.

1. Introduction

In today's ever changing business landscape, technology and innovation projects play a key role in creating competitive advantages for an organisation. However, many such projects are often hampered by under performance, cost overruns and lower than predicted revenue. This seems to indicate the lack of risk management in the way we manage projects. On the other hand, it is impossible to have any projects without risks. Thus, it is essential to have effective risk management rather than trying to eliminate risk out of projects.

2. What is risk management?

Cultural diversity can be, depending on the project, a source of high uncertainty, manifested in both positive and negative outcomes. Common project culture can also act as a helpful ingredient when managing the objectives of the project. The word risk has a long history and its meaning has been modified over time, causing debates about the correct terminology.

The management of negative impacts, or risks, has conventionally been stressed more intensively. Positive outcomes of uncertainty, or opportunities, and this type of thinking overall might not even be very suitable for traditional projects. When participants launch a project, they most likely have already estimated some benefits at least to be gained through the project. Some of the risks are also already known and proactive actions could have been taken, such as eliminating or minimising the impacts, transferring responsibility to another party, or actively accepting the consequences.

Uncertainty can be generated from external or internal sources. Uncertainty management is a continuous process. Both threats and opportunities could be managed simultaneously in the same process; only new risk identification techniques and response strategies are proposed to identify opportunities: exploit, share, enhance or ignore. The

impact of the same event might be positive to one project participant and at the same time negative to another.

Risk management involves the following steps (Fig. 1):

- Identify and analyze.
- Plan how to recognize, handle, and recover.
- Try to avoid or prevent.
- Mitigate occurrences.
- Recover software process.
- Need communication and reflection.

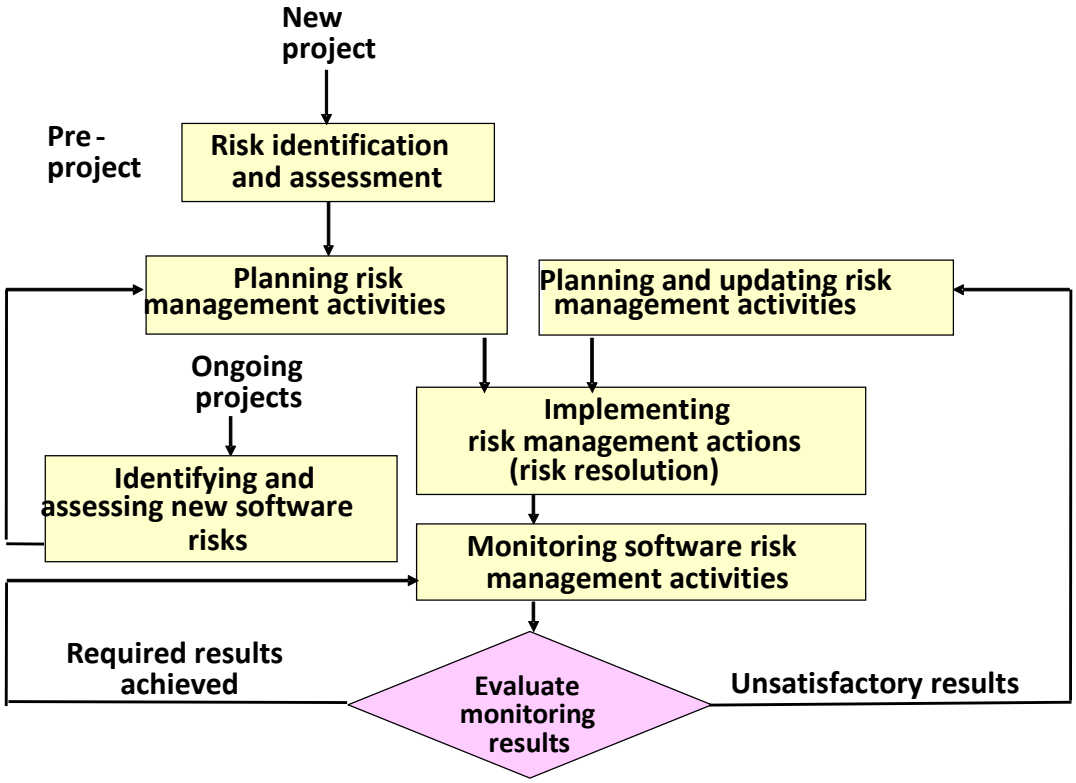

Fig. 1. The software risk management process

3. New criteria for effective collaboration

3.1 Some specific issues

When the knowledge base of an industry is both complex and expanding and the sources of expertise are widely dispersed, the locus of innovation will be found in multinational, rather than in individual firms.

In recent decades, there has been unprecedented growth in corporate partnering and reliance on various forms of external collaboration. The most common rationales offered for this upsurge in collaboration invorve some combination of risk sharing, obtaining

access to new markets and technologies, speeding products to market and pooling complementary skills.

Viewed broadly, technological change occurs in two forms. When advances build on existing know-how, established firms reap the bulk of the benefits. But when new discoveries create technological discontinuities, or radical breaks from previously dominant methods, incumbents can be robbed of many of their advantages.

Software development can be a high-value, but high risk undertaking. About 30% of software projects undertaken in the United States fail outright, and of the remainder, half finish with schedule and budget overruns that approximately double the original estimates (Standish Group, 1995). With the rise of outsourcing and off-shore development, many software development projects have acquired the additional risks that accrue to geographically distributed project teams – restrictive communication channels, differences of practice and policy, differences of language and culture, and time-zone challenges.

Many organizations now depend on collaborative work practices for their success. A collaborative work practice is a recurring process that can only be completed through the combined mental efforts of multiple people. Examples of collaborative work practices include mission critical tasks such as software requirements engineering, operational risk assessments, tender evaluations, and project proposal writing. While collaborative work practices can create significant value, collaboration can be a mixed blessing. Conflicts of purpose, unreliable information, poor communication, inadequate reasoning processes, and distractions can hamper a group's efficiency, limiting the value it can create. These difficulties become magnified when, as is often the case in the global economy, the contributors to collaborative work processes are separated by time and distance.

Technology, and in particular the development and deployment of new software systems for applications in government, business, industry, and entertainment, is one of the pillars of the globalizing economy. The software development process increasingly involves multiple teams, not only within a single organization, but across multiple organizations, and often across national, linguistic and cultural boundaries, motivated by both technical and business concerns. Technically, systems have become larger, more complex, and more interactive — especially over the internet and on the Web — and have a greater need for evolvability as a result of changes in the computer platform, the user community, and the application domain. Business issues involve not only cost-benefit tradeoffs and time-to-market concerns, but a need for high-quality technical and business services and for specialized application domain and development process knowledge.

All this drives the need to include sophisticated knowledge management techniques into collaborative software development. Knowledge management entails elicitation and specification, efficient and effective organization and access, knowledge generation, and abstraction, translation, and views. It also importantly requires knowledge protection, intersecting with intellectual property, privacy, and security concerns.

3.2 Collaborative knowledge

The handling and management of collaborative knowledge addresses three dimensions: the source of knowledge, the nature of the knowledge, and the use, application, and impact of the knowledge, particularly its effect on collaboration. In dealing with collaboration, it is useful to add another category to the standard distinction between explicit, implicit and tacit knowledge, namely whether the knowledge in question exists within individual partners or their people (for example, corporate practices), is common to all partners (scientific knowledge), is inherent in the collaboration (project

history), is publically available, or requires collaboration and integration to instantiate; a parallel measure deals with the use of the information — in individual organizations, or in the collaboration itself.

Table 1 presents an overview of the collaborative knowledge spectrum — sources and degree of awareness or concreteness.

	Level of awareness/concreteness				
		Partner			**Shared**
Source	**Managed**	**Explicit**	**Implicit**	**Tacit**	
Business & Technical Documents	Knowledge bases	Artifact libraries Policies History	Management goals/objectives	Document style preferences	Contracts Project history & configuration
Safety/Risk Information	RMMM plan	Safety procedures Risk assessment Triggers	Staff expertise	"Sixth-sense" awareness	Collaborative risks
Support services		Procedures Service contracts	Day-to-day conduct of service		
Tools and environment		Tools languages, notations, glossaries	Culture Idioms	Work practices	Shared infrastructure
Corporate environment		IT support Business network	Vendor/supplier decisions		History of collaboration
Expert knowledge		Publications White papers Memoranda Plans, etc.	Day-to-day processes Evaluation mechanisms Teaching	Social interaction Judgment	Professional contacts Modes of interaction
Legal knowledge		Documents	Interview & research strategies		International commercial agreements
External environment	External knowledge bases	News, statutes, regulations	Trends		(Inherently shared)
Collaborative		History Metadata	Communication Support	Trust	Protocols for interaction
Metrics & Inputs		Results & Trends	Relationships	Attitudes	Past & current projects
Social networking		Records of interactions	Social relationships		Implicit/tacit protocols

Table 1. The collaborative knowledge spectrum

4. Means for dealing with change

Flexible interfaces can provide support for business and management processes , including risk management , traceability and artifact flow , security , and knowledge management . They provide a natural place to hook:

- Policies for accessibility, transmission, and inter-component security.
- Policies and procedures for information hiding (whether for abstraction, access control, security, protection of intellectual property and privacy, or for more efficient and effective use in risk management), handling of revealed changes, or other management contingency policies.
- Translation of artifacts (language-to-language, notation-to-notation, perhaps glossary-to-glossary [motivated by standards and regulatory compliance across jurisdictions, as well as important national/regional variations in idiom]) and generation of abstract views.
- Selective propagation plus context-sensitive versioning of change information, artifacts, and possibly policies, depending on information about the context and status of the adjoining components and the interface itself.
- Generation of some types of summary artifacts, including schedule tracking.

They also provide a natural place for aspects of knowledge management , where various uses of information hiding also apply. Most of the above policies and actions have security and knowledge management implications.

In addition:

- Data mining of components—can use both pattern discovery in the component to find information, and filtering and abstraction, both to protect information and to hide obfuscating details—both during development and upon deployment/execution and maintenance/evolution.
- History and change information, component state, coherence of artifact configurations across components.
- Interesting question for discussion: ordering of filter-abstract-discover. Which leads to most patterns being discovered? Is there a chance of covert leakage?

Such interface objects will have references to the public state of the components it connects — not just object state, but process, project, product and business artifacts states.

Change is a key risk factor. The need to effectively accommodate it — whether a result of problems (corrective or preventive changes) or driven by changing needs and environments (adaptive or perfective changes) — has long been a major software engineering concern and is further exacerbated in large scale collaborative projects.

Table 2 outlines the change toolbox, a set of approaches that can be combined to tame and localize change. They are classified into process - and product - centric means.

Flexible high-level component interfaces allow for proper change localization, encapsulation and absorption of effects at component and partner boundaries.

Means for dealing with change		**Project Characteristics**	**Critical property when dealing with change**
Process-centric Means	Agility	Short iterations, team collaboration, customer involvement; change tolerance and flexibility, easier evolvability	High evolvability; relatively small, hierarchical or idiomatic project; tight scheduling constraints
	Traceability (artifact dependency	Easy to navigate traceability matrix; well-structured artifacts required to minimize dependencies; “immediate-automated”	Large-scale projects with clearly identified requirements; known requirements paired with need for high

	management)	identification of change impact; proactive monitoring of churn; tracking project progress	innovation; large number of diverse dependencies or complex dependence web
	Artifact flow management	Clearly identifies artifact types, partner responsibilities, and communication; basis for controlled artifact state and transformations	Large and complex project with diverse set of artifacts; large number of active stakeholders; complex flow of information
	Organizational Collaboration	Open organizational collaboration, open channels of information exchange, cooperative risk management, ease of change propagation via uniform and interconnected process views	Bidirectional component interaction; end-to-end product constraints or security, integrity, reliability requirements; high demand for information processing capacity; cross-partner domain expert collaboration
	Cooperative Risk Management	Clear roles and responsibilities of stakeholders when dealing with change; change taxonomy; hierarchical risk management plans	Large and complex project with diverse set of artifacts; large number of active stakeholders; complex flow of information; high level of innovation
	Knowledge Management	Identify, structure, and specialize implicit knowledge and relationships; identify implicit patterns (data mining); handle need-to-know, security and intellectual property	Domain uses implicit knowledge and/or subjective judgment; patterns of use matter; substantial intellectual property
	Domain Expert Collaboration	Identify critical cross-component domain issues, inconsistencies, and tradeoffs	Heavy use of domain knowledge; domain models or vocabulary variable or unstable
Product-centric Means	Software Architecture	Support for "plug-and-play" modular replacement, idioms to handle complexity and problems, easier change analysis and propagation via uniform structure; approach for cross-cutting concerns (e.g., aspects)	Hierarchical application; structural complexity; evolvability of behavior; service-oriented architecture with families of services/clients; inter-component cross-cuts
	Component Interfaces	Restricted propagation of change impact across component boundaries, reduced complexity of dependence web via information hiding; design for abstractability of dependence information	Partners responsible for different subsystems; complex or heavy information flow across boundaries; intellectual property or security concerns
	Interface Patterns	Flexibility of logical interfaces with fixed physical interfaces; access to legacy or COTS (commercial off-the-shelf)/GOTS (government off-the-shelf) services/databases; better information hiding at component boundaries; improved agility within components; support for refactoring within components	Collaborative development; negotiable boundaries between components; agile component development
	Adaptive Information	Propagation of content, system and bookkeeping information across component boundaries; safe extensibility of information semantics and component interfaces; improved agility within components	High component coupling; component-crossing extra-functional constraints (performance, timing, etc.); negotiable information flow

Table 2. Means for dealing with change ("Change Toolbox")

5. Conclusion

Software development has rapidly moved toward collaborative development models where multiple teams from different organizations and possibly companies, distributed across the globe, work together to define, develop, deploy and often collectively maintain complex software systems. To effectively accommodate such change product and process means, including flexible architectures and interfaces, iterative development, and cooperative, hierarchical risk management are required.

A key differentiator in collaborative software development is effective cooperative knowledge management — the process of acquisition, creation, and exchange of knowledge between all participants in the development efforts. This process is facilitated by the establishment of open culture and requires infrastructure services that support it.

References:

[1] *Risk-Driven Management Contingency Policies in Collaborative Software Development*, International Journal of Information Technology and Management, 2009.

[2] *Risk Management for Collaborative Software Development, Information Systems Management*, 25 (4), 20–30, Fall 2006. Reprinted in The EDP Audit, Control, and Security Newsletter, 35 (3), March 2007.