International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014
ISSN 2091-2730

Software Development Risk Management Using OODA Loop

Sanjeev Kumar Punia, Dr. Anuj Kumar, Dr. Kuldeep Malik
Ph.D. Scholar, NIMS University, Jaipur, Rajasthan - INDIA
puniyasanjeev@hotmail.com
+91 999 919 0085

ABSTRACT - Software development projects are subject to risks like any other project. These risks must be managed in order for the project to succeed. Current frameworks and models for risk identification, assessment and management are static and unchangble. They lack feedback capability and cannot adapt to future changes in risk events. The OODA (Observe, Orient, Decide and Act) loop, developed during the Korean war by fighter pilot Colonel John Boyd, is a dynamic risk management framework that has built in feedback methods and readily adapts to future changes. It can be successfully employed by development teams as an effective risk management framework, helping projects come in on time and on budget.

KEYWORDS - OODA loop, risk management, dynamic risk management, requirement risks management, software mitigation risks, unanticipated risks, futuristic risk assessments.

INTRODUCTION - Software development projects are subject to risks like any other project. Software development is subject to unique risks which can be mitigated through effective risk management techniques. Risks are unavoidable and must be managed. Successfully managing risks assists developers in completing the project on time and on budget. Strategies selected to manage risk may result in a better product than originally anticipated. Identifying, analyzing, tracking, and managing software risk aids crucial decision making including release readiness.

The fighter pilot Colonel John Boyd developed a series of four steps that he noticed fighter pilots followed during air to air combat Korean War. These four steps are observe, orient, decide and act that is known as the OODA loop. Col. Boyd went on to become a superb fighter pilot and Pentagon strategist. Current risk management frameworks are static and unchangeable as well as they lack feedback capability and cannot adapt the future changes in risk events. The OODA loop is a dynamic risk management framework that is built with feedback methods and readily adapts to future changes. Software development teams can employ the OODA Loop to manage risks reduction that affects their projects.

LITERATURE REVIEW - Software development projects are not immune to risks. Risk management strategies are crucial to identify, track and reduce risks. The software's spend shows that only 2% of software was able to used as delivered by the study of Department of Defense (DoD) in 1995. 75% was either never used or cancelled prior to delivery. Cook *et. al.* [1] explained that $35.7 billion spent on software management and much research involves surveying current software developers with program manager professionals. The similar result is found by using different strategies to identify, track and reduce risk. The similar components of risk were identified by reviewing past research experience. Ropponen *et. al.* [2] explained that the risk components include scheduling risks, timing risks, system functionality risks, subcontracting risks, requirement management risks, personnel management risks as well as resource usage and performance risks.

The knowledge lacking of software suppliers adds an increased level of risk. Schinasi *et. al.* [3] stated that it poses a large problem to DoD as they mostly contracts it out and does very little of its own software development. Risks starts from changing requirements, lack of skills, fault technologies, gold plating and an unrealistic project schedule. According to Suresh Babu *et. al.* [4], gold plating developers develop a better requirement beyond the objective. Mohtashami *et. al.* [5] explained that, the development teams spread across a building, the country or even the world as companies grow. Distributed development teams add risk to software projects as they are not in a centralized location. According to Borland Software Corporation [6], collecting the requirements from stakeholders is very important but more important than that is continuing to request requirement, analyze and specify requirements to eliminate redundancy and avoid unnecessary risks. Cook *et. al.* [1] explained that requirements elicitation, analysis, documentation, verification, review, approval, configuration control and traceability should be incorporated into sound risk management procedures.

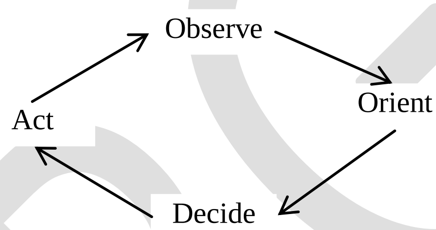
The identification and planning is the best way for risk reduction early in the development cycle. Leonard *et. al.* [7] explained that software development and inspections focus to avoid risks before introducing them into the project. The time, money and effort are used during the development process to mitigate the risks before beginning. Jørgensen [8] suggested that an increased identification of risks led to an over confidence and over optimism in estimating software development efforts. Stoddard *et. al.* [9] explained that company history, structure, processes and reward systems can facilitate the risk management process. The conceptualize requirements is a popular method for tracking, identifying and managing the requirement risks. Various model based requirement management approaches exist for better identification, tracking and managing requirements. In the past, models were not formally connected to software development such that there was no way to ensure programmers design decisions used in the mode.

Uzzafer [10] stated that a lot of factors as project characteristics, risk management team, risk identification approaches and project quality contribute and affect the level of project risk. Assessing the impact of project risk and residual performance risk provide a better understanding of effectiveness and adequacy for risk management techniques. The risk management capabilities play important roles in managing software projects either implemented in any fashion.

However, the conceptualization and development of risk management theories lags the requirements of practice. Bannerman [11] found in research studies that risk management practice lags the understandings of risk management such that current frameworks and models for risk identification, assessment and management are static and unchangeable. They lack feedback capability and cannot adapt to future changes in risk events. Sarigiannidis *et. al.* [12] stated that dynamic risk management frameworks provide futuristic assessments of risk events by coupling with static models that can enhance the project success. The software development projects are greatly benefited from model based requirements engineering as identifying, assessing, analyzing, verifying, tracking and managing requirements that reduce risk to software projects. A big research is not conducted that relates the OODA loop for risk management in the software development process. This work concerned mainly with agile software development. Steve Adolph [13] relates the OODA loop to agile software development and argues that agility depends on the tempo that iterate through that loop. The development speed depends on culture but not on methodologies or tools used.

This paper is primarily an introduction to the OODA loop and agile software development. It briefly outlines the fitting of OODA loop with the notion of agile software development and proposes research opportunities. Ullman [14] explained the use of OODA loop to business and product development. He specifically explained the get stuck of business and product development teams where action never occurs. He also explained that the guidelines to unstick the OODA loop for making decisions and taking action.

Colonel John Body's Loop - Colonel John Boyd was an air force fighter pilot and brilliant military strategist in United State. During the Korean War, Boyd observed a cycle of four actions that pilots took during combat and named these actions OODA loop. He explained that pilots with OODA loop are faster than others dominate dogfights. The pilots without OODA loop forces constantly re-observe and re-orient themselves. OODA loop prevents the pilot from making decisions and taking action to gain the upper hand. The OODA loop is composed of four steps: observe, orient, decide and act as shown below.



The OODA loop

FIGURE 1

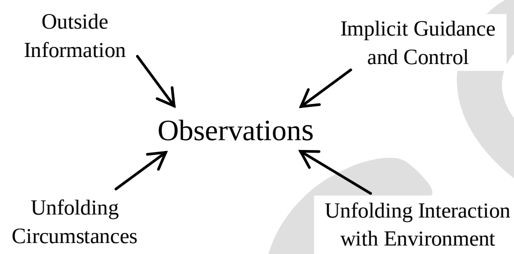
Development teams cycle through these steps repeatedly. In OODA loop, observation phase deals with collection of data for situation and surroundings. Orientation phase is the analysis of data to form a mental perspective. The decision phase chooses a specific course of action based on gathered and analyzed data. Action phase is the physical act of executing the decision. The results of the action should be observed and the cycle repeats till the completion of the requirements.

Although OODA loop created for air to air combat fighter pilots but it applies to risk management for software development also. As fighter pilots apply the OODA loop to manage risk in combat same way stakeholders, project managers and developers apply the OODA loop for prevention of crash and burn in software projects. The OODA loop also assist to manage scheduling and timing risks, system functionality risks, subcontracting risks, requirement management risks, resource usage and performance risks as well as personnel management risks.

The OODA Loop and Software Development Risk Management: Observe - The first step in risk management is to identify or observe the risks so failing to identify risks can drastically harm software projects. There are four factors that influence observations in the OODA loop that include outside information, unfolding circumstances, unfolding interaction with environment and implicit guidance with control. These factors are external to the loop and assist developers and project managers with risk identification by combining them. Outside information is required for effective risk management. Software developers must receive absolute information from stakeholders because eliciting requirements from stakeholders is time consuming.

International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014
ISSN 2091-2730

Early care should be taken for identifying all classes of stakeholders from all involved organizations. The development can be complicate after missing a stakeholder requirement. The unfolding circumstances can change the risk posture in requirement identification, design, development or test during development. One source of requirements creep is the failing of capture requirements during the requirements identification phase. The cost and effort increases in integrating new requirements as development progresses. Even more costly is to fix the bugs as development progresses.

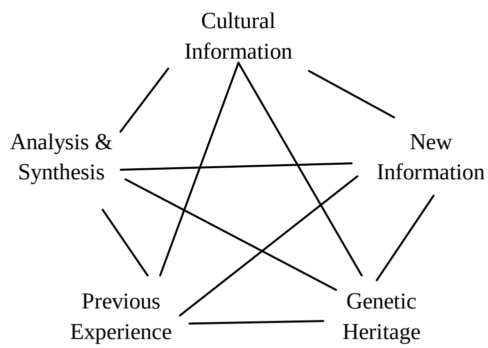


The observe step of the OODA loop

FIGURE 2

Apply the OODA loop on a small scale is a good practice when new requirements or even coding bugs are identified. The development teams must ensure the working of each component works as intended during the development progresses and components completion. The components may have unexpected consequences during interfaces. These side effects may be mitigate through careful planning and design. Interaction with the environment is also critical in developing software for a system. The OODA loop is feed with implicit guidance and control at each stage. It is especially crucial during observation to identify and plan for direct orders, key performance parameters, laws and regulations.

Orient - The input of orient phase is the generated information from the observation's first step of the OODA loop. The orientation aligns observed information into a well defined, logical manner to take decisions more readily. During this stage, risks must be assessed based on probability of occurrence and the potential impact. Based on calculated composite risk indices many risks can be ranked. The severity of the risk is proportional to composite risk index. Col. Boyd identified five factors those contribute the orientation of the pilots based on observed information. These five factors include cultural traditions, new information, analysis and synthesis, previous experiences and genetic heritage. The relationships between the five factors are shown below.



Factors influencing the orient step of the OODA loop

FIGURE 3

The data, requirements, systems and circumstances change that leads to new information the team can use to identify and orient the project to manage risks. This factor is actually a mini observe step built within orientation step. It is a reminder for teams to constantly absorb new information and watch for unanticipated risks. The analysis and synthesis is a no brainer while information and observations are useless without analysis. Analyze the identified risks allows teams to determine appropriate and effective risk management techniques. Software can be analyzed for functionality, bugs and completeness that can be synthesized and tested. The three factors as cultural traditions, genetic heritage and previous experiences are very similar to each other for risk management of software development projects.

The cultural traditions refer to the culture and traditions of the organization. The team or organization may have a preference for one software development or requirements model. The genetic heritage describes the management of projects and risks for developers and stakeholders in software project. Development teams and project managers rely on previous experiences to identify and manage risks in current projects. The past successfully or unsuccessfully completed project risks may affect current projects also so team members use past project experience to understand the tracking and mitigating of current risks.

Decide - The development team chooses the risk management strategy after identification and analysation of risk with orientation of project goals. Additional implicit guidance, risk action plans and contingency plans turns into a problem during this phase. The risk can be decrease greatly by the ability to identify, analyze, monitor and track requirements and project status through the development lifecycle. It is much easier to plan and integrate requirements at the beginning of the software development cycle. The cost and effort to implement new requirements, makes changes and fix bugs increases as software development progresses. Feedback from decisions flows back to the observation step. The risk management strategy chosen may affect the project schedule or budget and it may change although it might accomplish the same function. The benefit of effectively managing requirements is based on the working of software feedback occurrence when decisions are made that allow quick iteration through the OODA loop.

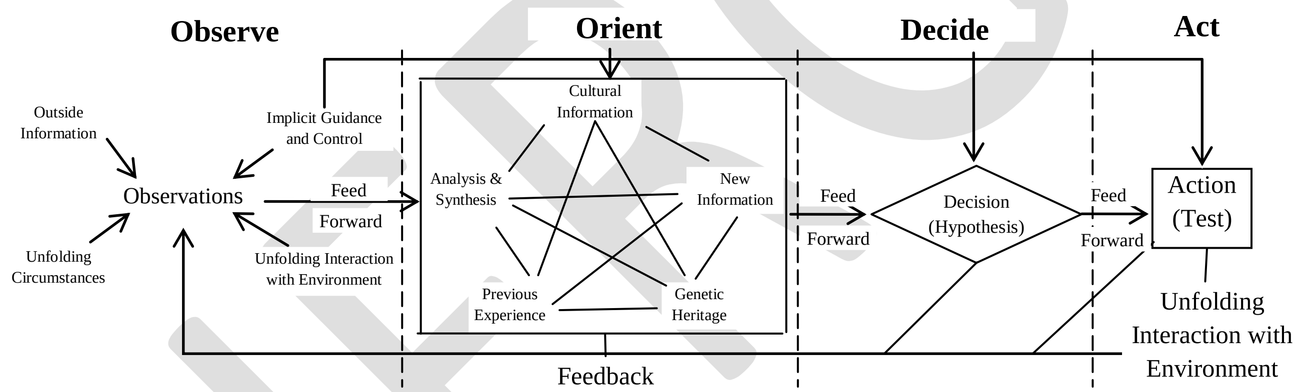
Act - Execute phase starts after the decision of risk management strategies. The risk management is not a quick process but risks can be managed by quick fed back in the observation stage. In this we have to check the operation of software

International Journal of Engineering Research and General Science Volume 2, Issue 6, October-November, 2014
ISSN 2091-2730

components with it indentation and further to check the development to tweak the software or correct bugs that ensure stakeholders happiness with the progress and results. The additional risk is added as requirements begin changing and requirement creep sets. The risks should be tracked and the results of the risk management strategies recorded and shared. Team members and stakeholders need to know the status of their project. Another iteration of the OODA loop is performed if the risk is not reduced as anticipated.

The loop - All combined four steps with influencing factors are shown below. The OODA loop is not a once through framework for risk management and it applied repetitively throughout the entire software development cycle. Risks remain in the project development till project completion. Teams should not get stuck into observing and orienting to risks and development status and must be decide and act on observed information.

The decision and action stages feedback flows back to the observation stage. Continually observe the results of team decisions and actions. Repetitive iterations of the OODA loop will reduce software project risks and increase the likelihood of completion on time and within budget.



The full OODA loop as described by Col. John Boyd

FIGURE 4

CONCLUSION - The OODA loop is a tool for effective risk management like all projects. Software projects also have the risk so software project teams can use the OODA loop as a risk management framework. Each step helps developers and project managers to identify, track and manage risks. Due to the cyclic nature of the OODA loop, multiple iterations can be applied to the project as risks evolve over time. Successful implementation of the OODA loop assists project managers in completing their projects within budget and on time. We plan to use the OODA loop as a risk management framework for a software project in the future. We will try to test its effectiveness over the course of the project. Each identified risk will be tracked and all observations, decisions and actions will be recorded.

REFERENCES:

[1] D. A. Cook and T. R. Leishman "Requirements risk can drown software projects" Journal of the Quality Assurance Institute, vol. 17, no. 2, pp. 56 - 64, 2012.

[2] J. Ropponen and K. Lyytinen, " Components of software development risk: How to address them? A project manager survey" IEEE Transactions on Software Engineering, vol. 26, no. 2, pp. 86 - 94, 2010.

[3] K. V. Schinasi "Defense acquisitions: Knowledge of software suppliers needed to manage risks" United States General Accounting Office, Washington D.C., 2009.

[4] G. N. K. Suresh Babu and S. K. Srivatsa, "Increasing success of software projects through minimizing risks" International Journal of Research & Reviews in Software Engineering, vol. 1, no. 1, pp. 20 - 22, 2011.

[5] M. Mohtashami, T. Marlowe and V. Kirova "Risk management for collaborative software development" Information Systems Management, vol. 23, no. 4, pp. 21 - 28, 2012.

[6] Borland Software Corporation "Mitigating risk with effective requirements engineering" Available: http://www.borland.com/resources/en/pdf/white_papers/mitigating_risk_with_effective_requirements_engineering.pdf, 2012.

[7] J. G. Leonard, T. R. Adler and R. K. Nordgren "Improving risk management: Moving from risk elimination to risk avoidance" Information and Software Technology, vol. 41, no. 1, pp. 28 - 35, 2013.

[8] M. Jørgensen "Identification of more risks can lead to increased over-optimism of and over confidence in software development effort estimates" Information and Software Technology, vol. 52, no. 5, pp. 504 - 512, 2010.

[9] J. Stoddard and Y. H. Kwak "Project risk management: Lessons learned from software development environment" Technovation, vol. 24, no. 11, pp. 914 - 919, 2011.

[10] M. Uzzafer "A new dimension in software risk management" World Acadamey of Science, Engineering and Technology, vol. 64, pp. 341 - 343, 2010.

[11] P. L. Bannerman "Risk and risk management in software products: A reassessment" The Journal of Systems & Software, vol. 81, no. 12, pp. 2117 - 2121, 2008.

[12] L. Sarigiannidis and P. D. Chatzoglou "Software development project risk management: A new conceptual framework" Journal of Software Engineering & Applications, vol. 4, no. 5, pp. 294 - 308, 2011.

[13] Steve Adolph, “What lessons can the agile community learn from a maverick fighter pilot?,” In Proceedings of the Agile Conference, Vancouver, BC, pp. 98 - 102, 2006.

[14] D.G. Ullman “The sound of a broken OODA loop” Crosstalk: Journal of Defense Software Engineering, vol. 20, no. 4, pp. 21 - 24, 2013