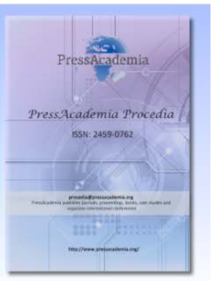
PressAcademia Procedia - (PAP), ISSN: 2459-0762

PressAcademia Procedia



Global Business Research Congress (GBRC), May 26-27, 2016, Istanbul, Turkey.

MANAGING INTERACTING SOFTWARE PROJECT RISKS

DOI: 10.17261/Pressacademia.2016118662

Pradip Peter Dey[1], Muzibul Khan[2], Mohammad Amin[3], Bhaskar Raj Sinha[4], Hassan Badkoobehi[5], Shatha Jawad[6], Laith Al Any[7]

[1,3,4,5,6,7]National University. pdey@nu.edu mamin@nu.edu bsinha@nu.edu hbadkoob@nu.edu sjawad@nu.edu lalany@nu.edu

[2] Kyocera Communications. muzib100@gmail.com

ABSTRACT

Managing risks in a software project can be challenging. There are many risk categories including communication risks, project planning risks, technical risks, budget risks, scheduling risks, legal risks, ethical risks, operational risks, security risks, and personnel risks that require timely attention. Potential risks should be identified, analyzed and evaluated. Appropriate strategies should be developed for managing imminent risks in a timely manner. This paper advocates a strategy that assigns a special role to communication risk, because it interacts with other risks in a way that may allow its coupling with most other risks and changing their effects. Interacting risks may have to be studied at a higher level with special attention to communication risk, because it could be considered as a super-risk. In order to get best results with the current state of knowledge, a team of risk analysts may provide early warnings about potential risks that are then further studied in interaction contexts for developing appropriate management strategies. By integrating risk based project management into a software development process, coordination of all activities in a comprehensive manner would be possible with a special emphasis on interacting risks.

Keywords: Communication, iterative models, risk analysis, software development, volatility.

1. INTRODUCTION

Considerable progress has been made in the study of software project risks since 1980's (Boehm 1991; Charette 1989; U.S. Air Force Systems Command 1988). However, additional studies may help software project managers in understanding interactions among software risk factors for applying modern management approaches to the problem of risk management. Most experts would agree with the opinion that "the popular interest in management as a discipline and a field of study is fairly recent. But management, both as a practice and as a field of study, has a respectable history, in many different countries, going back almost two centuries." (Drucker 2008, p.12). Modern approaches to management developed in the U.S.A. after World War II are currently taught in colleges and universities. The four classic management functions one may learn in school are "planning, organizing, leading, and controlling" (Nelson & Economy 2005, p. 5). At a level of details, one may argue for adding to the list more functions and skills such as communicating, decision making, meeting ethical standards, analytical thinking, coaching, listening, negotiating, and visioning. Project management challenges can be viewed from a process perspective and many experts such as McFarland (1970) highlighted this view. Management is the "process by which managers create, direct, maintain and operate purposive, organizations though systematic, coordinated, cooperative human effort" (McFarland 1970, p. 5). "Because projects are transient, their delivery follows a development process, from germination of the idea, through initiation, design and delivery, to commissioning, handover to the client and closeout of the work." (Turner

2014, p. 5) However, in addition to process phases, certain aspects of the nature of a project may pose formidable challenges that require special attention. Henry Gantt who was a pioneer in planning and control techniques proposed a chart, popularly known as the Gantt chart, as a project management tool (Gantt 1919). Software projects have benefitted from Gantt's insights and other innovations. However, software projects have often failed even under standard supervision of process elements and with standard management techniques (Leffingwell & Widrig 2000). That is, in addition to traditional project management functions, there are additional challenges for software project management, which play significant roles in modern software industry. The scholarly nature of software analysis, modeling, algorithms, model checking, mathematical precision, recursivity and their combined effects elevate the problem solving knowledge level beyond the reach of ordinary managers. The intellectual nature of software is different from traditional products (Armour 2015). Managers experienced with traditional projects may suffer from the burdens of the past and fail to appreciate the nature of complex interactions among elements of executable knowledge (Armour 2015).

American philosopher, psychologist, and educational reformer John Dewey told us "We do not learn from experience ... we learn from reflecting on experience." What we learn from our reflection on our experience with software projects is that many projects fail due to mismanagement of risk factors. "The Standish Group research shows a staggering 31% of projects will be cancelled before they ever get completed. Further results indicate 52.7% of projects will cost 189% of their original estimates." (Leffingwell & Widrig 2000, p.6). Most software projects start in a hurry with a limited understanding of the problem to be solved with an assumption that a better understanding would be achieved after further analysis of the problem with appropriate resources. However, volatility of the project elements starts appearing with considerable ambiguity as further analyses are carried out. Various aspects of software projects interact in a way that is hard to understand and difficult to manage. Various risk factors may become clear to some team members or specialists in the process of detailed analysis. However, communication of such risk analysis may pose another level of risk. That is, communication risk is a super-risk that may negatively affect other risks both due to the difficulties involved in the process of communication and more importantly because it permeates all aspects of software development. For example, if the problem has un-decidable aspects that require technical communication at mathematical levels then the communication difficulties may prevent a general understanding of the problem among the team members and stakeholders. Other interactions among risks need to be studied in a new innovative way in order to make improvements in examining consequences of risks and integrating risk analysis properly into the center of software project management. Interacting risks should be studied using a comprehensive model of interactions among risks, process phases, software components etc., and properly integrated into the project management framework. The goal of this paper is to review the current practices of software project management and address software project management challenges by integrating software aspects with agile and iterative models of development using risk analysis and management strategies. There are many risk categories including communication risks, technical risks, budget risks, scheduling risks, legal risks, ethical risks, operational risks, security risks, and personnel risks that require timely attention often from specialists. Potential risks should be identified, modeled, and evaluated. Appropriate strategies should be developed for managing the imminent risks in a timely manner. This paper advocates a better-shared understanding of software complexity through studies of interacting risks together with software component interactions and interactions among software process phases. A team of risk analysts may provide early warnings about potential risks that are then further studied for possible interactions in order to develop appropriate management strategies. By integrating risk management into a software development process, an innovative approach needs to coordinate all activities in a comprehensive manner. It should place special emphasis on interacting risks, which play prominent roles in most commercial systems.

2. LITERATURE REVIEW

"Risk is our business." (Star Trek: Return to Tomorrow 1968). Risk is a factor in every endeavor, and mitigating risk factors requires knowledge, planning and resources. Risk management is an essential component of every technology-based project. Many software projects have failed in the past due to challenging management issues. Some researchers have paid attention to these challenges and made progress towards a better understanding of them (Barros, Werner & Travassos 2004; Boehm 1991; Chemuturi & Cagley

Jr. 2010; Jones 1998; Gulla 2012; Keil, Cule, Lyytinen & Schmidt 1998; Xia & Lee 2004). Some of the studies assessed the problem as follows “. . .software has long been one of the most troublesome technologies of the 20th century, and one that has long been resistant to executive control. One of the main reasons that software is difficult to control is because it has been difficult to estimate software projects, and to measure software quality and productivity in an accurate way” (Jones 1998). In September 2015, the U.S. Environmental Protection Agency (EPA) disclosed that Volkswagen could face fines of as much as $18 billion for allegedly manipulating exhaust-emissions tests. The EPA has initiated a criminal investigation against the company. “Volkswagen acknowledged that it installed software in some diesel-powered cars to make it appear that the cars met tough U.S. anti-smog rules, the EPA said.” (The Wall Street Journal, Sept. 22, 2015, page B1). Michael Horn, Volkswagen’s U.S. chief executive told a House subcommittee hearing on October 8, 2015 that a deceptive piece of software was put into as many as 11 million Volkswagen vehicles worldwide in order to fool emission-testing equipment (Los Angeles Times, Oct 9, 2015). A study conducted by West Virginia University’s Center for Alternative Fuels, Engines and Emissions, or CAFEE, established that in normal driving conditions, “nitrogen oxide emissions – one of the top six common air pollutants – from two Volkswagen light-duty diesel engines exceeded the EPA’s Tier 2-Bin 5 standard. One vehicle exceeded the standard by a factor of 15 to 35 and the other by a factor of 5 to 20” (West Virginia University Today 2014). Volkswagen’s deceptive software was designed to sense when the car was being tested and then activated equipment that reduced emissions in order to pass the test. The software turned the equipment down during normal driving, “increasing emissions far above legal limits, most likely to save fuel or to improve the car’s torque and acceleration. The company may also have been worried that some parts of the emissions system would break down if used continuously” (New York Times 2016). The management of ethical risk factors failed in this case and unethical conducts did not surface due to lack of proper communication. Structured and unstructured communication among software developers, managers, analysts, and other stakeholders usually plays an important role in identifying risk factors in a timely manner and mitigating the risks. Risk assessment and risk control should be part of every software project (Jalote 2002).

From experimental studies of software projects, we have learned about potential risk factors and their mitigation by taking timely actions (Boehm 1991; Charette 1989; Jones 1994). An apparently good approach is to identify the risks early, and take remedial steps quickly. According to a study by Keil, Cule, Lyytinen and Schmidt (1998), three independent panels of experienced software project managers, “selected a common set of 11 risk factors as being among the more important items” (Keil, Cule, Lyytinen & Schmidt 1998: p. 78) for risk based software project management. The study emphasized identification of risks and their relative importance as perceived by managers. This is necessary but not sufficient for averting project failures. What is required is a set of effective steps against the risks so that their cumulative effect can be stopped in a timely manner to save a vulnerable software project. Lack of top management commitment to the project is often cited as an important risk factor although it was not clear as to whether robust communication channels among the stakeholders were present in the environment. Risks may proliferate rapidly in an interactive spiraling manner with harmful consequences when communications among stakeholders degrade.

In a recent study, “insufficient communication” was identified among the prominent factors responsible for project failure (Gulla 2012). In a careful review, Gulla (2012) observes that by properly classifying documented causes of IT project failure, 54 percent are attributed to poor project management - the most cited factor, whereas only 3 percent are attributed to technical challenges. Gulla (2012) identifies seven key factors that are primarily responsible for project failures: (1) poor project planning and direction, (2) insufficient communication, (3) ineffective management, (4) failure to align with constituents and stakeholders, (5) ineffective involvement of executive management, (6) lack of soft skills or the ability to adapt, (7) poor or missing methodology and tools.

In order to understand the role of various key factors, experimental data may be collected from software project manager and other stakeholder. Often subjective questions are asked to project managers or interested stakeholders and responses are statistically analyzed in order to come up with a perceived account of the problem. Xia, and Lee (2004) tried to relate their account of information system development project complexity with project performance based on responses from subjective questions. Other studies (such as

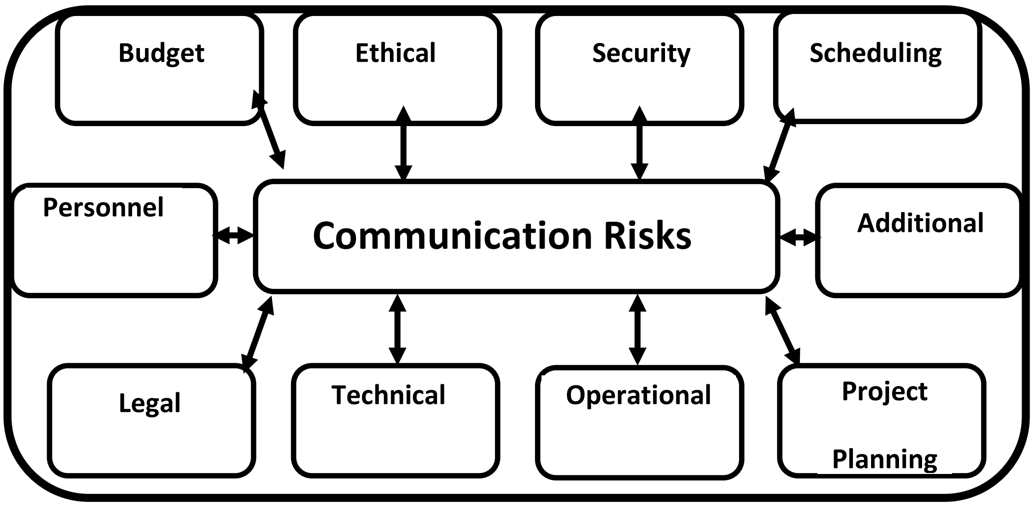
Keil, Cule, Lyytinen, & Schmidt 1998) attempt to account for risk analysis based on surveys with subjective questions. Considerable progress has been made with these studies; however, a better understanding of software development projects is needed in order to make additional improvements. Software development is different from other traditional product development. “Rather than a product in the traditional sense, software is better viewed as a container for the real product. What the customer buys and the user employs is the executable knowledge contained in the software” (Armour 2015, p.32). The true nature of software imposes an overarching requirement on the study of the interactions of risk factors with the preeminent role of communication risk. Software development has often been considered as one of the most challenging processes of modern technology. A complex software system with highly interactive elements is difficult for dynamic modeling and allows multiple interpretations and development perspectives (Pressman & Maxim 2014; Sommerville 2015). Some approach it from a scientific perspective while others treat it in an artistically creative manner. Over the decades, a multitude of approaches to software development have been proposed. Donald Knuth (1969) initially indicated that software writing is an art (Knuth 1969). David Gries argued it to be a scientific endeavor (Gries 1981). Watts Humphrey invested a considerable amount of time and effort in the study of software development problems and proposed software development primarily as a process (Humphrey 1989). In recent years, many practitioners have studied the software development from a different point of view and have come to realize that software is engineered (Pressman & Maxim 2014; Sommerville 2015; Braude & Bernstein 2011). Because of the adoption of engineering methods along with agile or iterative process, there is an opportunity to address software’s project management challenges in a new way. An analytical account of communication risk and its interactions with other risk factors along with interactions among software components and process phases may reveal certain aspects of software projects and their contributions to the project management. An analytical account will set the stage for further experimental studies that are required in order to make progress in software project management.

2.1. Interactions among Risks

Software systems are among the most complex artifacts humans ever built. An analytical account of the software components and the project risk factors in their interaction context is needed in order to make further progress in understanding complexity of software systems. The communication risk and its interactions with other risk factors will be specially emphasized here because of the complexity created by these interactions. Armour’s (2015) view of software reveals some important aspects of software that pose a serious challenge to the traditional concepts held by most software project managers. Software project managers should be prepared to deal with collaborative knowledge development where structured and unstructured communication among the developers plays the most important role. A strong team of software professionals would be motivated to develop a shared understanding of the problem (Li, Ko, & Zhu, 2015). In order to achieve the shared understanding, professional software developers often depend on intuitive extensions of meanings of words, phrases, terms, gestures, diagrams etc. in semi-structured or unstructured communication environments. Limited terminologies, diagrams, languages and artifacts that are available today can be found in popular software engineering textbooks (Pressman & Maxim 2014; Sommerville 2015; Braude & Bernstein 2011). Additional artifacts can be found in IEEE and ISO standards, journals, monographs and reference manuals such as Rumbaugh, Jacobson and Booch (2005). However, despite all these, software engineers struggle in finding contextually meaningful terms, phrases, languages, tools and artifacts in order to succeed in communication. Recent studies have often identified inadequate or insufficient communication as one of the most important factors for project failures or setbacks (Gulla 2012). Poor project planning and direction is at the top of the list of factors and an important part of planning is to assign the right people to the right task and make clear assignments to team members, with defined goals and responsibilities (Gulla 2012). It is reasonable to argue that communication factor interacts with project planning and direction with resulting dynamics that evade analysis if their combined effects are not adequately studied. Suppose the project under consideration has considerable security risk and two team members need to be assigned security related responsibilities with resulting dynamics of three way interactions. The lack of adequate terms and artifacts makes it difficult to communicate many aspects of security while the general problem of security is proven to be un-decidable (Harrison, Ruzzo & Ullman 1976). The main argument is that the communication problem in software projects interacts with other problems in such a way that many software projects suffer substantial loss resulting from the interactions. That is, communication risk is a super-risk that may negatively affect other risks for a considerable time period because of the difficulties involved in the process of communication. In the case of Volkswagen's recent problem mentioned above, many suspected risk factors including technical risks, legal risks, ethical risks, operational risks, and personnel risks etc. might have been drastically impacted by the communication risk leading to an unprecedented debacle. Although the Volkswagen emissions cheating scandal was an ethical failure, it could only happen in an environment of inadequate corporate governance, bad corporate management, and bad software project management. Failure in corporate oversight by definition generally boils down to a failure in communication, as does failure in management.

Every software project plan should include aspects of timely communication. Communication framework should be based on detailed aspects of meeting times or frequencies, meeting minutes, participant lists, minutes distribution lists, share-point sites, reviews, approvals etc. Ethical risk and other risk factors interact with communication in such a way that software team culture should be built around a model pragmatically dominated by the role of communication. Such a model is presented in Figure 1 for visual inspection. This model is intended to help software engineers and other stakeholders in the practical aspects of interactions of risk factors with a dominant role of communication risks. The model shows that every risk category is related to every other category where communication risks play a central role.

Figure 1: Interactions among Software Project Risks



The above interaction model should be understood in the context of software development processes. Certain agile and iterative software development processes have incorporated frequent structured and unstructured communication events that are often effective in the face of the challenges resulting from the interacting risks. In these development models, the lack of proper terms, artifacts etc. are often eventually overcome by means of informal intuitive and extended communication among team members who work closely with each other in a process model that frequently put them on communication in a collaborative environment. If a software project manager is an active participant in such a team, then communication risk and other interacting risk factors might be mitigated effectively in a timely manner. Since software projects deal with creation of executable knowledge (Armour 2015), there is an extra demand on most of the active participants to rise up to a level of intellectual maturity where communication can take many forms using a wide range of formal and informal artifacts, tools, terms, languages etc. If the teams are separated geographically and culturally, the communication challenges may get harder to manage, and establishing a feeling of trust and belonging is also difficult in distributed teams (Carmel & Agarwal 2001; Holmström, Conchúir, Ågerfalk, & Fitzgerald 2006;

Espinosa, Slaughter, Herbsleb & Kraut 2007). Careful monitoring of risks and periodic review of the impacts of interacting risks is essential. From the preceding logical analysis, it is clear that (1) interactions among risk factors, (2) interactions among distributed team members, (3) lack of adequate tools, languages and artifacts of communication, (4) interactions among software components, and (5) interactions among software development phases all together make the issues more complicated where success depends heavily on a robust system of communication that should be fully developed in order to deal with the future software systems.

3. CONCLUSION

Software development involves highly creative activities marked by a wide range of complex communication patterns among software engineers and stakeholders in order to make joint decisions in a cooperative manner, often at an intellectual level that is not well understood. It is evident from the published studies that software project management challenges are different from those of traditional project management, because of the nature of software, which is very different from traditional products. In this context, the logical importance of communication risk and its relationship with other risk factors as it permeates all facets of software development lead us to consider it as a super-risk, a special category for its volatile interactions with other risk factors and software elements in patterns that are often considered to be "beyond the reach" of ordinary humans. Well-coordinated team efforts by well-formed teams are needed in order to deal with interacting risks in complex software projects. That is, while individuals may find the challenges of managing complex software projects overwhelming, a strong team of professionals may overcome the challenges by integrating appropriate management strategies into an iterative software development process in a collaborative manner. Some iterative development models are flexible enough to accommodate the special role of communication risk within their processes allowing a broad range of communication forms including semi-structured, un-structures, formal, informal, intuitive, structured and unstructured. Future studies may be suggested with the goals of collecting experimental data on the nature of interactions of among risk factors in order to validate the model proposed in this study or finding a more appropriate one.

REFERENCES

Armour, P. G., (2015). The Business of Software: Thinking Thoughts, Communications of the ACM, (Vol.58, No. 10, pages 32-34).

Barros, M., Werner, C. & Travassos, G., (2004). Supporting Risks In Software Project Management, Journal of Systems and Software, Volume 70, Issues 1–2, Pages 21–35.

Boehm, B., (1991), Software Risk Management: Principles and Practices, IEEE Software 8,1, pages 32-41.

Braude, E., & Bernstein, M., (2011). Software Engineering: Modern Approaches, (2nd Edition), John Wiley & Sons.

Carmel, E., & Agarwal, R., (2001). Tactical Approaches For Alleviating Distance In Global Software Development, IEEE Software, Vol. 18, No. 2, pp. 22-29.

Charette, R.N., (1989). Software Engineering Risk Analysis And Management, Intertext, New York.

Chemuturi, M. & Cagley Jr., T., (2010), Software Project Management: Best Practices, Tools and Techniques. J. Ross Publishing, Plantation, Florida.

Drucker, P., (2008). Management (Revised Edition), Harper Collins, New York.

Espinosa, J. A., Slaughter, S. A., Herbsleb, J. D., Kraut, R. E., (2007). Team Knowledge and Coordination in Geographically Distributed Software Development, Journal of Management Information Systems, Vol. 24, Issue 1, 2007.

Gantt, H., (1919). Organizing for Work, Harcourt, Brace, and Howe, New York.

Gries, D., (1981). The Science of Programming. Springer, 1981.

Gulla, J., (2012). Seven Reasons IT Projects Fail, IBM Systems Magazine, Retrieved August 8, 2015 from: http://www.ibmsystemsmag.com/power/Systems-Management/Workload Management/ project_pitfalls/

Harrison, M. A., Ruzzo, W. L. & Ullman, J. D., (1976). Protection in Operating Systems, Communications of the ACM, Vol. 19, No.8, Pages 461–471.

Holmström, H., Ó., Conchúir, E., Ågerfalk, P.J., & Fitzgerald, B., (2006). Global Software Development Challenges: A case Study on Temporal, Geographical and Socio-Cultural Distance, ICGSE2006, Costao do Santinho, Florianopolis, Brazil, Oct 16-19, 2006.

Humphrey, W., (1989). Managing the Software Process, Addison-Wesley, Reading, Mass.

Jalote, P. (2002). Software Project Management in Practice. Boston: Addison Wesley.

Jones, C., (1994). Assessment and Control of Software Risks, Prentice-Hall, Englewood Cliffs, N.J.

Jones, C., (1998). What it Means to be 'Best in Class' for Software, Capers Jones, Software Productivity Research, Inc. www.spr.com.

Keil, M., Cule, P. E., Lyytinen, K., & Schmidt, R.C., (1998). A Framework for Identifying Software Project Risks, Communications of the ACM, Vol.41, No. 11., pages 76-83.

Knuth, D.E., (1969). Seminumerical Algorithms: The Art of Computer Programming 2. Addison-Wesley, Reading, Mass.

Leffingwell, D. & Widrig, D., (2000). Managing Software Requirements: A Unified Approach, Addison-Wesley, New York.

Li, P. L., Ko, A. J. & Zhu, J., (2015). What Makes A Great Software Engineer?, The 37th International Conference on Software Engineering, May 20-22, 2015, Florence, Italy.

Los Angeles Times, (2015). VW bosses say they didn't know, Los Angeles Times, October 9, 2015, (page C1).

McFarland, D., (1970). Management: Principles and Practices (3rd edition),The McMillan Company, London.

Nelson, B. & Economy, P., (2005). The management bible, John Wiley & Sons, Hoboken.

New York Times, (2016). Explaining Volkswagen's Emissions Scandal, Reported by Guilbert Gates, Jack Ewing, Karl Russell and Derek Watkins, Updated April 22, 2016. Retrieved April 24, 2016 from:http://www.nytimes.com/interactive/2015/business/international/vw-diesel-emissions-scandal-explained.html

Pressman, R. S. & Maxim, B., (2014). Software Engineering: A Practitioner's Approach. (8th edition), McGraw-Hill.

Rumbaugh, R., Jacobson, I, & Booch, G., (2005). The Unified Modeling Language Reference Manual. (2nd Edition), Addison Wesley.

Sommerville, I., (2015). Software Engineering, (10th edition), Addison Wesley.

The Wall Street Journal, (2015). U.S. Begins Criminal Probe of VW, Reported by William Boston, in The Wall Street Journal, Sept. 22, 2015, page B1.

Turner, R., (2014). A handbook for project management practitioners, in Turner, R. (ed.), 2014, Gower Handbook of Project Management, Gower Publishing, Burlington.

U.S. Air Force Systems Command, (1988) "Software Risk Abatement," AFSCIAFLC Pamphlet 800-45, 1988.

West Virginia University Today, (2014). WVU study found elevated levels of emissions from Volkswagen vehicles, September 24, 2014. Retrieved April 12, 2016 from: http://wvutoday.wvu.edu/n/2015/09/24/wvu-study-found-elevated-levels-of-emissions-from- volkswagen-vehicles

Xia, W., & Lee, G., (2004). Grasping the complexity of IS development projects, Communications of t ACM, Vol.47, No. 5, pages 68-74.

Acknowledgement: Authors are grateful to Debra Bowen, Arun Datta, Abdur R. Amin and many others for their comments and suggestions on earlier versions of this paper.