

**Managing Global Software Development Teams:
Technology and Policy Proposals for Knowledge Sharing**

by

Satwiksai Seshasai

**Bachelor of Science, Computer Science and Engineering
Massachusetts Institute of Technology, 2001**

**Master of Engineering, Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2002**

Submitted to the Engineering Systems Division
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Technology and Policy

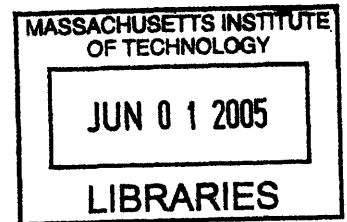
at the

Massachusetts Institute of Technology

June 2005

©2005 Satwiksai Seshasai.
All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly
paper and electronic copies of this thesis document in whole or in part.



Signature of Author.....
Technology and Policy Program, Engineering Systems
Division
May 5, 2005

Certified by.....
Prof. Joel Cutcher-Gershenfeld
Executive Director, Engineering Systems Learning Center
Thesis Supervisor

Accepted by.....
Dava J. Newman
Professor of Aeronautics and Astronautics and Engineering Systems
Director, Technology and Policy Program

ARCHIVES

**Managing Global Software Development Teams:
Technology and Policy Proposals for Knowledge Sharing**

by
Satwiksai Seshasai

Submitted to the Engineering Systems Division on May 5, 2005
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Technology and Policy

Abstract

This thesis uses an in-depth case study, with integrated data analysis, to compare and contrast globally distributed and co-located software teams within the IBM Corporation. Important differences in information sharing, collaboration and other behaviors were observed, along with a range of innovations in work operations. Technology and policy implications that draw on the benefits of each model are identified.

The development of this thesis began with a seminar that was conducted at MIT to invite stakeholders in various areas of knowledge-based offshore outsourcing to discuss strategic, economic, organizational and technical issues raised in various environments. Large and small firms, various industries, and various business models were covered. The context provided by these stakeholders was used to design an in-depth case study at IBM, with the focus on a matched pair of software teams, which were studied for a period of one year. Both software teams were identical in aspects such as product scope, team size and domain; however, they differed in the key aspect that one team's members all work on the same hallway while the other team's members are geographically dispersed among multiple locations in the United States and India

Quantitative technical data from the source control system of each team, the software problem report database, frequency and content of group emails, weekly meetings, and individual interviews were combined with qualitative data from stakeholder interviews to distinguish key benefits and challenges of each model. The quantitative measures gauged data such as frequency and methods of collaboration, social and technical networks, and differences in handling strategic and tactical decisions. The qualitative interviews discussed stakeholder perceptions of the quantitative data, and their motivations for decisions related to knowledge sharing.

Key findings from the data include a number of observations about specific forms of knowledge sharing which differentiate the two teams. The distributed team used electronic mail as a forum for discussion which peaked around project deadlines, while the collocated team relied on e-mail as an announcement mechanism. Team meetings for the distributed team were much more tactical and task oriented in nature than meetings of the collocated team. With regard to the technical project itself, developers on the collocated team shared source code to a much greater extent, however status input to the software problem report database was much more interactive on the distributed team.

This thesis is also important for pioneering highly precise indicators of team interactions based on the coding of archival data derived from e-mail, telephone, meeting and other

interactions. The methods developed hold great promise for further studies of design teams, as well as a feedback tool that could be highly valuable for these teams.

A number of emerging themes were found in the data analysis, which suggest that lessons from this study need not only apply to cases where geographic distribution is a factor. The teams consistently showed that the same technologies, processes and stages of the project lifecycle can be handled very differently based upon context. Also, social relationships and dominant individuals on a team can have an impact on technical productivity. Finally, the evidence in this case suggested that geographic structure need not define destiny, and in some cases geographic structure can be used as an asset.

The data analysis points to preliminary technology policy implications at the individual, team, organization, and national levels. At the individual level, it is recommended that workers in distributed teams alter work hours to devote a few minutes after-hours to synchronous communication with team members in different time zones – something that happened more often among the members of the co-located team. On the other hand, in a collocated team, it is recommended that the team use software tools to discover technical expertise that is more formally recorded among the members of the distributed team. At the team level, specific added value gained unintentionally from one geographic structure – such as greater documentation of decisions on a distributed team – can be achieved in co-located teams. At the organizational level, this thesis provides methods for assessing an organization's tacit knowledge capital at a much more granular level than tabulating patents or licenses. A number of institutions such as corporate training and development departments, labor unions, professional associations and government education and training initiatives may be impacted by the changes to workforce training and work methods suggested by this thesis. At the national level, lessons from these teams demonstrate that the drivers for policy decisions related to offshore outsourcing need to be adapted in knowledge-based industries which have the potential for globally shared tasks, and export regulations dealing with intellectual property exchange in global software teams need to account for the daily trade of IP in geographically distributed teams.

As the thesis focused on one in-depth case study, a significant effort is made to propose future research directions which can validate the proposals with broader data collection.

Thesis Supervisor: Prof. Joel Cutcher-Gershenfeld
Executive Director, Engineering Systems Learning Center

Table of Contents

Acknowledgements.....	6
Introduction.....	7
Overall Research Question	10
Hypotheses	11
Specific:	11
Source code	12
Group E-mail	13
Software Problem Reports (SPRs).....	13
Team meetings	13
Informal Contact	13
Theoretical Foundations – Literature Review.....	14
Participant – Observer Method of Research in Software Development	14
Global software management	15
Offshore outsourcing	17
Global manufacturing	20
New Product Development.....	22
Cultural study in software organizations	23
Methodologies.....	26
Context Development	26
Research Subjects	27
Quantitative Data	30
Personal interviews	30
Software Problem Reports	31
Weekly Meetings over 1 year – coding system for tasks, status	32
Source Code Control System.....	33
Group Email Exchanges	34
Qualitative Data	34
Personal Interviews before data analysis	34
Personal interviews after data analysis	36
Participant – Observer model of research.....	36
Stakeholder Analysis From Expert Seminar.....	38
Demand Management	39
Long-term Productivity.....	40
Integrated Value Chain	41
Organizational Models.....	42
Other Decision Factors	42
Common Barriers within Firm.....	43
Location Choice	43
Linking the Seminar to the Data Gathering	44
Core Findings.....	46
Structural Determinants of Coordinated Development.....	46
Using Electronic Mail for Asynchronous Discussion.....	47
Technical Collaboration through Shared Source Code.....	54
Nature of Team Meetings	58
Using Technology to Update Work Item Status.....	62

Informal Communication: Impact on Trust, Creativity and Problem Solving	68
Emerging Themes	72
Same Technologies and Processes can be Applied Differently.....	72
Social Relationships and Technical Productivity are Intertwined	73
Teams React Differently to Different Stages of the Project	74
Structure Does Not Define Destiny	75
Individuals can Heavily Influence Coordinated Development.....	76
Geographic Distribution can be an Asset.....	77
Technology and Policy Impact Assessment	78
Individual	78
Team	79
Organization.....	81
Institutions.....	82
Nation.....	82
Future Research Directions.....	85
Larger Number of Teams.....	85
Results of Weekly Feedback to a Team.....	85
Collection of Other Data	85
Varying Types of Software Teams	86
Other Knowledge-Based Industries	86
Technology Prototypes	87
Conclusion	88
Appendices.....	91
Research Prospectus Submitted to IBM	92
Interview Protocol Submitted to IBM.....	93
References.....	94

Acknowledgements

This thesis would not have been completed without the help of many individuals. First and foremost, Prof. Joel Cutcher-Gershenfeld's guidance on both content and structure helped to turn a set of disorganized thoughts and goals into a solid contribution to the field. He inspired me to appreciate the value of understanding organizations through his class, and in a matter of months taught me how to conduct my first real social science study. However, I would have never reached the point of being able to conduct this study without the years of counsel from Prof. Amar Gupta. His insight into knowledge management across borders and his personal attention to my academic and professional success over the past four years have been invaluable. I am also grateful for Prof. Lester Thurow's support for the seminar which was the foundation of this study. Indeed, the breadth of TPP's curriculum provided valuable guidance in this endeavor.

IBM played an especially valuable role in this process, as the company both funded my individual academic program and served as the subject for this study. Merle Braley, Sheryl Jablonowski and Bill Hume were especially supportive and encouraging in my academic endeavors, and I thank them for providing the support and serving as a proponent of my studies throughout the organization's approval processes. Colleagues on my team who participated in this study were especially helpful with their time and thoughts – in any participant-observer study it is challenging to separate roles and my colleagues made it a very comfortable process.

I would like to thank my parents, who instilled within me a work ethic and appreciation for academic pursuit without which I would not be nearly as successful, and then provided gentle support along the way. Finally, my fiancée Rebecca deserves more gratitude than I could ever provide, for giving me love and encouragement through all of the late nights working. Her promise to marry me as soon as I finish this thesis provided all the motivation I needed.

Introduction

The field of software has matured to a point where large software organizations need to treat the engineers, infrastructure and technology used to manufacture software as an engineering system. This systems treatment of the software field involves an understanding of the many new aspects which are introduced when a software organization is made global. These aspects include cultural assumptions about workload, process and quality, political uncertainty in the various participating nations, technical infrastructure requirements of a temporally and globally distributed manufacturing chain – both to share knowledge and source code, and financial uncertainty which could affect team size and location.

This case study looks at two distinct teams at one of the world's largest software companies – IBM. The two teams are similar in virtually all structural, organizational and technical aspects, but differ in the geographic location of team members – one team is entirely co-located while the other is geographically dispersed. The goal of this study is to identify potential methods for large software development organizations to *leverage* these social, technical, cultural and organizational aspects of their system of engineers, rather than taking these aspects as fixed assumptions in attempts to globalize the software organization. An example of this is using data from the source code change management system could be combined with informal social network data from the interviews and cultural aspects of work and communication style to yield more flexible and effective collaborations between the engineers. Such examples suggest that a systems model which characterizes the relationships between various aspects of a global software engineering system can expose methods of increasing the various forms of output prioritized by different organizations such as schedule, quality, and volume.

Existing studies in this field have used survey methodologies to gather information about the individuals and teams involved; statistical data analysis of source code change management, time to resolution of tasks, and quality measures; and information technology to build tools to bridge the gaps between dispersed workers. A key characteristic of such works such as the study by Herbsleb et. al. of Lucent Technologies¹ and a case study by Haag, Foley and Newman² is the treatment of a distributed software development system as essentially a co-located software development organization with the added *challenge* of distribution. Thus, the methodologies developed in the literature on distributed software development provide a valuable means for characterizing the salient features of software development organizations and assessing relationships, performance and output. However, this field will benefit from treating software engineering as a system in the manner described above, and not considering the distribution as a challenge to overcome while keeping all other assumptions about the system constant. Another common factor in the literature is that outputs such as delay and speed (studied by Herbsleb et. al.) should be studied on the task level rather than the project or team level. However, studying the software development process as a system can allow software engineers to assess the overall impact on their portfolio of tasks. For example, one conclusion that could be reached is that even though an individual task may take longer to complete in a distributed environment, the 24-hour engagement on tasks allows for a greater volume of tasks to be completed in a given project cycle.

This thesis is organized into the following sections:

- **Research Question:** A discussion of the specific research goal of this study.
- **Hypotheses:** General hypotheses which were tested in this study, along with specific postulations regarding each of the specific types of data collected.

- **Literature Review:** A comprehensive look at the foundational works which were used to devise this study.
- **Methodologies:** A description of the seminar which was conducted to uncover salient research questions, the specific quantitative measures which were taken over the one year period, and qualitative interviews which were conducted with team members.
- **Stakeholder Analysis:** Relevant factors uncovered in the expert seminar, which drove decisions around which data to collect in the case study.
- **Core Findings:** Results from the data collection methods, both quantitative and qualitative, plus a set of emerging themes.
- **Technology and Policy Impact Assessment:** A series of proposals motivated by the findings, which focus on increasing levels of generality from the individual to the nation.
- **Future Research Directions:** As this case study was very limited in scope, a number of research directions were discovered which could generalize the results of this study to knowledge-based industries.
- **Conclusion:** A broader context for the study as well as reflections from the participant-observer who conducted this research.
- **Appendices:** In addition to relevant appendices related to this particular case study, other short works which have relevance to the broader context of this study are included.

Overall Research Question

The overall goal of this research is to determine the relative benefits associated with globally distributed and the co-located software team, both on the project and product, as well as how technology and policy solutions can be built to translate advantages from each model to the other. The focus is on benefits because much of the literature was found to focus on challenges, and treat geographic distribution itself as a challenge. The project refers to the people, processes and organizations involved, while the product refers to the actual technical output of the project. There is an explicit interest in both the project and the product because this may help to drive more dynamic solutions which make changes to the project structure to impact the product. In the context of this research question, we define technology and policy solutions as those which combine changes in the technology (both technology being used to support the project and technology being produced in the product) and changes in the policy (both organizational policy and public policy). Thus, the research question is quite broad – as it seeks to understand a very broad context – however the goal and the question is very simple: Are there distinct benefits associated with co-located and globally distributed software development teams and, if so, are they applicable or adaptable across both situations?

Hypotheses

This study was undertaken with a general hypothesis on the nature of knowledge sharing in global software teams, and a set of specific hypotheses related to each of the forms of communication being studied. Because it is an exploratory study, with a detailed year-long look at a small sample, the work serves as much to help generate and clarify hypotheses as it provides preliminary tests of these propositions. A full testing of these hypotheses would involve follow-up research with a larger sample.

Overall, the hypothesis which motivates this thesis is that the relative benefits of each form of software development – co-located and dispersed – have the potential to be translated to the other form, if they are properly understood and explicitly acknowledged. The two teams which were studied in this report were used to understand the knowledge sharing which occurs in each form, and then understand the potential for translating these knowledge-sharing methods between forms. For example, while a co-located team may thrive on informal in-person encounters, a dispersed team can also find ways to encourage informal interaction, especially when significant business value can be derived. This translation of knowledge-sharing methods leads to policy implications on every level – from individuals to teams, organizations, nations and the world – because it suggests that the traditional rules which govern the transition of knowledge work offshore may not apply. Finally, this study also examines the hypothesis that technology used to drive various knowledge-sharing methods must be well understood and well utilized in order to achieve the level of translated methods discussed.

Specific Methods of Knowledge Sharing:

As components of the above general hypotheses, specific hypotheses related to the particular knowledge-sharing methods studied in this report were examined as described below.

The areas can be divided into three categories: inputs, processes and outputs. In this study we controlled for almost all inputs except team geography, used a similar set of processes for each team, and judged the outputs as well as the implementation of the specific processes. The system of inputs, processes, and outputs is illustrated below:

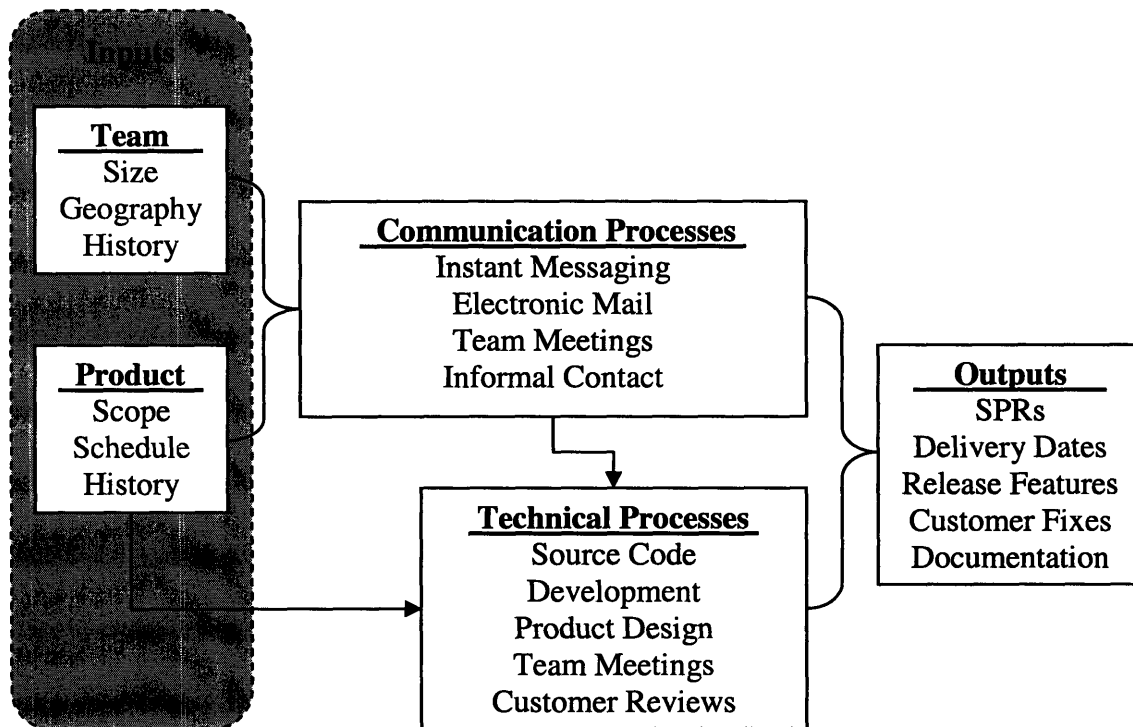


Figure 1: The Inputs, Processes, and Outputs involved in the system of software development are shown. The team and product characteristics are put through a series of communication processes which both generate input into the technical process and also directly generate outputs. The product inputs can also be directly applied to the technical processes.

Source code

The hypothesis here is that source code is "less collaborative" in a distributed team because of lower interaction between team members. Modules are kept to single developers or distributed geographically, possibly without an intentional design decision to do so. Reciprocal check-ins – the modification of the same module simultaneously by different developers – is lower in a distributed team.

Group E-mail

The hypothesis here is that group email is more relied upon in a distributed team. Threads are longer in a distributed team, while similar issues get resolved in other manners on a co-located team. Also, the average number of developers per thread is higher in a co-located team because a co-located team only relies on group-wide discussions if the issue requires input from the entire team.

Software Problem Reports (SPRs)

The hypothesis here is that the average time from the SPR's Approved state – when the SPR is approved to be fixed – to the Resolved state – when the SPR is fixed – is the same for both teams. This is because SPRs are a core function of both teams, and since both teams are generally successful at their function, this metric should remain almost equal.

Team meetings

The hypothesis here is that team meetings are more strategic in nature for the co-located team, while they are more tactical in nature for the distributed team. There is more developer to developer contact in the distributed team at the team meeting itself, while in the co-located team, this contact takes place outside the meetings. More specific tasks are assigned in the distributed team's meetings.

Informal Contact

The hypothesis here is that there is more informal contact on the co-located team, which leads to solutions that would not have otherwise been reached as quickly. On the distributed team, other forms of communication are used, but the informal nature of these communications is not as high.

Theoretical Foundations – Literature Review

Participant – Observer Method of Research in Software Development

The most relevant literature used in developing this research project was research on software development done in the same participant-observer method that was used in this study. Each of the works described below was used to build a specific component of the data gathering process. As a whole, this work builds on each of the works discussed below, and the specific ways in which this is done are described as well.

Seaman³ presents a comprehensive model for participant-observer research in software teams. Elements from Seaman's model which were used in this study include the importance of coding qualitative results; the notion that unlike other participant-observer studies, in software it is necessary to spend more time with meeting minutes and electronic mail exchanges because participants' thoughts are not as readily obvious from their work; and the notion that a participant-observer should not cause the other participants to change their behavior because they are being observed. While Seaman's participant-observer model refers to an embedded observer rather than an observer who is actually a software developer, Poltrock and Grudin⁴ discuss participant-observer studies where the observer is a full-fledged member of the software team. One of the concerns they cite is the researcher's subjectivity based on an investment in the project – in their study this was limited by ensuring that the participant-observers did not focus on critical parts of the project and thus had less of a stake in the final product. While this was not possible with our study, the balance between loss of objectivity and richness of data gathering was acknowledged. Meyer⁵ and Burns and Vicente⁶ describe this balance as one which favors participant-observation in cases where the researcher is able to spend enough time to familiarize themselves with the domain and also with the social dynamics of the subject group.

In our case, as the researcher was a member of the group for two years prior to conducting the research, this was indeed the case. Myers⁷ points out another benefit of participant-observer research: the ability to question what an outsider may be forced to take for granted since they do not have the depth of understanding of the participants. This was used as motivation in conducting the data analysis, as many assumptions about the software development environment were questioned. Finally, Elliott and Scacchi⁸ used the participant-observer method in an open source development community, where there was no strict organizational structure binding together the participants. They found that this technique enabled them to gain trust and build community in a way that other research methods could not. In gathering the experiences of other participant-observer studies as described above, it became clear that one model which is not traditionally used is for a participant whose primary role is a software developer to transform into the observer – rather than the other way around. This is the model which is used in this study, with the intention of building on previous software industry participant observer studies.

The remaining sections of this literature review cover related sub-components of the study – since part of the contribution of this report is the actual methodology for studying knowledge sharing on software teams, an extensive literature review is useful.

Global software management

The teams studied in this paper are global, and thus it is relevant to understand the issues which have been raised in the literature regarding global software management. In general, the literature points out a need to place special attention on the way a software project is managed when the team is global, and this is done in our study through the qualitative interviews as well as the inspection of weekly meeting minutes which describe task allocation. The specific issues raised in the literature on global software management are below.

Taylor and Bain treat the call center as a "white collar factory", and incorporate employment relations concepts from the factory⁹. The style of software development discussed in this paper requires the same focus on employment relations because the tight interaction between employees means that if there is disparity in employee relations between sites, it will manifest as a significant issue. Talent is the central component of knowledge-based economies, and thus talent management should not be outsourced and should be treated as a core requirement for success¹⁰. Aron and Singh note that IT work is on a knowledge continuum – and information workers play a key role at each stage¹¹. However, in managing a software team, it is important to note that management practices in America can't always be transferred to other countries, so we need to understand the cultural values involved as well¹². Finally, the 24-hour knowledge factory being proposed is presented in the context of software new product development. Johnson notes that similarities exist between toys and software with respect to changing customer demand and short product lifecycles among other factors, and that toy manufacturers outsource to allow risk management of new products¹³.

Managing the knowledge in a 24-hour software factory is obviously an important issue, and research into knowledge management in the offshore model is useful in providing a framework. Barthelmy suggests that managing the sourced effort is the largest hidden cost – and organizations should keep their core knowledge within the core of the company¹⁴. Lei and Slocum, on the other hand, argue that alliances between sites must understand and share core competencies, not be "deskilled"¹⁵. Tallman addresses the need to take advantage of knowledge networks around the world using capability based theory where the process can act as leverage¹⁶. Powell states that knowledge-creation is the key core competency for firms, and collaboration is key to this creation of knowledge¹⁷. Continuing this point, Davenport notes that managing

customer support knowledge is vital – it is impossible to replace this direct, human input with automated tasks¹⁸. Companies that succeed will have methods for managing the knowledge within their firms to ensure that duplication of effort is eliminated¹⁹. The 24-hour knowledge factory model furthers this ideal by bringing those working on the same tasks onto the same teams, ensuring that knowledge is shared between those who need it shared the most.

Offshore outsourcing

In addition to simply understanding global software development, it was also useful to review the general literature on offshore outsourcing without specific focus on the software industry. This was used in preparing the qualitative interviews, because it allowed the discussion to consider potential aspects of the work environment which were not specific to software development. It was also useful to review this literature in producing the technology and policy recommendations which came from the data analysis because the data collected was focused on one particular team and the literature provided useful context in which to consider the data.

Agrawal et. al. describe “round-the-clock shifts” offshore, where some firms even pay higher wages for offshore workers to work odd hours²⁰. Their research has found that companies can reduce costs 30 to 44 percent for many types of work including R&D by performing “round-the-clock” shifts. However, these shifts all take place in the same offshore location, rather than passing tasks between locations as suggested in our model. In choosing a model, Kaka presents a spectrum of 6 models for offshore partners: supplemental staff, turnkey projects, assistance in building centers, build-operate-transfer, assets, and joint ventures²¹. This framework was useful in demonstrating that the 24-hour model is the next natural step in this progression.

Many researchers have described emerging facets of offshore outsourcing that suggest a need for 24-hour knowledge factory models. Saunders et. al. cite technical capability as a greater

driver than cost savings, and suggest maintaining core functions onshore to preserve this technical capability²². Barney cites transaction cost economics as the only factor in determining whether to keep tasks within company boundaries²³. Carr suggests that offshore models can't just modularize tasks, and need to build strategic competencies in all locations for all tasks²⁴. Three factors always exist for competitive advantage, according to Christiansen - economies of scale and scope, integration and non-integration, and process-based core competencies²⁵. Light indicates that managers must understand cultural values of themselves and their employees in all locations to be successful²⁶. DiRomualdo and Gurbaxani provide more guidelines for assessing outsourcing: improvements to information systems, business impact, and generation of new revenue²⁷. In placing the offshore outsourcing thrust in a national context, Young states that the U.S. must try harder to compete globally, because the U.S. can't give up their standard of living²⁸.

Once the 24-hour knowledge factory is developed, organizational issues faced by offshore outsourcing firms will need to be understood. Workforce and rework dynamics make outsourcing challenging, according to Mizoras - thus, organizations need third-party firms whose competency is purely in building organizations²⁹. Champy describes X-engineering, a new model which is changing organizational relationships by introducing transparency, standardization, and harmonization³⁰. Organizational learning is cited as one major factor in building flexible offshore models³¹. Lacity et. al. agree that organizations should focus on continuous learning more than the strategic offshoring versus commodity debate³². Furthermore, they state that firms should concentrate on efficiency, not costs, when choosing an outsourcing location³³, and the maturity of the technology should be part of the decision on what to outsource³⁴. Quinn suggests strategically outsourcing so that a company chooses each location based on core competencies –

one implementation of the 24-hour knowledge factory involves a similar notion because each task in the organization has access to each location, so particular competencies for particular location can be exploited to the maximum³⁵. Fuchs notes the need to align product-market focus, resources and capabilities, organizational culture, direction – this is a challenge to be faced with many global centers, especially when working on the same task, as one virtual organization³⁶.

The GLOBE study demonstrated that cross-cultural teams need to be managed with an appreciation for the specific cultural drivers which exist in each individual culture³⁷. Emerging markets need to be treated differently than developed markets in terms of marketing strategy– this is a cycle, where as the markets develop, learning will come from them and the strategies will change³⁸.

The multi-shoring concept is offshoring to multiple locations, based on skills availability and competencies³⁹. However, it does not consider the case when those multiple locations can be used to share tasks, as in the 24-hour knowledge factory. Millman advocates moving beyond commodity outsourcing, and using outsourcing to transform processes⁴⁰, a motivation for the 24-hour model proposed here which is not possible without outsourcing. Cheifetz says to outsource strategically, and beware of cultural differences which could hinder it⁴¹.

With regard to the value chain and decisions on what is best to outsource, research has conflicted on whether outsourcing the innovation, or new product development, is best. Chesbrough states that outsourcing innovation negative if done between countries because it wont be successful if conflicts of interest exist⁴². However, Quinn states that outsourcing innovation is a means for companies to succeed – he suggests that outsourcing needs to take advantage of talent at all stages of the value chain, and so new product development can be done in offshore locations⁴³.

Organizations need to understand the “ecosystem” within which they exist, and consider the health of the organizations around them which affect their performance⁴⁴ - in a 24-hour knowledge factory, the ecosystem of the software development environment becomes much broader, and the ecosystem concept applies at the team level. This involves a global mind-set and not being dependant on single country or culture factors for making business performance decisions⁴⁵.

Kumra states the need to distinguish between activities which require proximity to the customer and activities which can be done remotely, and then distribute tasks appropriately⁴⁶ – in our model, a given task can have a customer-facing component as well as a remote component and take advantage of both since the tasks are shared between locations, with no need to make a decision between the two. In this complex environment, relationship management is the key to outsourcing ventures, because the situation almost always changes from what was initially planned⁴⁷.

Finally, outsourcing requires management styles which involve negotiating results rather than issuing orders⁴⁸. In a 24-hour knowledge factory, this notion needs to be taken one step further, in that the factory must allow the employees in each location to negotiate tasks with each other, since the coordination required does not allow for a centrally managed system.

Global manufacturing

The literature on offshore outsourcing in the previous section focused largely on knowledge based work, and on taking work that is done in the United States and performing it offshore on a case by case basis. To understand issues involved in developing long term captive centers, it became useful to look into literature on global manufacturing. Since the software development process includes an aspect of manufacturing – the source code is packaged and

delivered as a product by the software developers – some of the aspects related to manufacturing globally are relevant. In this study, the Indian team members were part of a major company site located in India, and thus issues of plant location, overhead costs and manufacturing process – while not directly discussed in the data collection – had an impact on the team.

Many lessons in global plant location can be taken from the manufacturing industry, which has been building globally distributed factories for a much longer period than the knowledge-based industries. Lovelock points out that manufacturing lessons can be applied to services, by understanding global drivers such as location choice⁴⁹. Rosenfeld et. al. highlight the need for plant location decisions to be based on more than cost – strategic considerations such as local skills are important in building more flexible and efficient plants⁵⁰. These concepts are useful in the 24-hour knowledge factory model because outsourcing is traditionally done for cost savings, but more strategic considerations are necessary if the outsourcing is being done in the model proposed in this paper. Capability focused plant location is used in manufacturing, with three factors being important decision criteria: complexity, diffuseness, and well-developed interfaces⁵¹. Locations should be seen as value centers, with the specific source of value for the particular center being explicitly identified⁵². The four sources of value are service, investment, cost and profit centers. Since new product development requires each of these value centers to be present, the 24-hour knowledge factory model provides a means for multiple locations, each with their own value center, to be incorporated in the supply chain for the product. In a globally distributed factory environment, information management is important for entire lifecycle of the product⁵³. Thus, in building a global knowledge factory, companies can't just cut overhead costs as they would in a typical offshore environment; instead, they need to transform manufacturing processes for long-term success⁵⁴. Pisano points out that knowledge-based companies can't just

invest in innovative R&D and outsource manufacturing – they need to also invest in manufacturing process⁵⁵. In software development, where the manufacturing is done by the individual workers, the 24-hour knowledge factories can be seen as a means to innovate the manufacturing process.

New Product Development

In designing this study, one of the decisions which was made was to focus on new product development. Other areas of software – such as maintenance release development, customer fix development, and solutions development – have differences in inputs such as team size, product scope and project schedule. Since we are focusing on new product development, the literature on this was useful to review.

In new product development, research has shown many factors to be relevant to a successful delivery model. Coordination between locations is key (not decentralization), and configurations are different for firms such as Boeing and others who use multi-national models for new product development⁵⁶. Earl presents risks for distributed new product development such as the loss of organizational learning and innovation⁵⁷. Granstand et. al. cite the need to manage technological competencies throughout a distributed organization, and put focus on diffusing technological competencies between groups, especially in new product development where technological competencies may have different results in different product groups⁵⁸. At Dell, technology was used to innovate the coordination between different parts of the supply chain so that all organizations were treated as one company⁵⁹. Basically, the model used by Dell suggested that technology can be used to transform a vertical integration into a more horizontal framework while maintaining the structure of the vertical model. This is the same concept which

drives the 24-hour knowledge factory to maintain “plants” in various locations, but share tasks horizontally between locations.

Cultural study in software organizations

Inherent in all aspects of global software teams – from team communication to product features – is the issue of culture. Different ethnic cultures being represented on a team can lead to communication and work process differences, as well as differences in how the team members choose to implement certain features. In this study, we asked specifically about the cultural impact on the project and the product, when interviewing team members. For this reason, it was useful to review the literature on cultural study in software organizations.

Much of the research on global software teams describes the cultural differences between team members as a challenge that needs to be overcome. A study at Sharp Laboratories cited the key struggle in software development management as between features, schedules and resources and described the challenge in coordinating these three factors between developers in the United States, Japan and India⁶⁰. According to the study, the best practices that were common from one culture were near impossible to implement in the other cultures because of the base assumptions of the team members. Another study which followed the experiences of a U.K. software company’s attempt to open an office in Bangalore, India concluded with the findings that the cultural factors should have been better prepared for at the outset of the project⁶¹. Many aspects of Indian culture, such as personal responsibilities of developers, importance of processes and measures of success, and factors which motivate were all not appreciated by the U.K. managers. One telling quote from the study that describes the ignorance states: “I have books on coaching methods, motivation, etc. which don’t seem to work here. I would have thought that with all the Western movies, TV, etc. that this would have brushed off on the Indians.” The assumptions of

this study will be refuted in part later in this paper because the argument that management style simply has to be adapted for local context does not allow for the possibility that management styles can take into account strengths from different cultures on the team and form a hybrid approach between forcing Western culture and accepting Indian culture.

Some works move closer to the model proposed by this paper, suggesting sensitivity towards different cultural assumptions and emphasizing cross-cultural environments rather than homogeneity. Walsham describes three case studies in which cross-cultural teams working in information and communication technology (ICT) faced cross-cultural problems⁶². In these studies, he defines culture as “shared views” that deal with beliefs, power systems, behaviors, and structural values. He develops a systematic means for assessing culture on these dimensions, and then provides examples of cultural insensitivity which leads to failed communication – one example is an Asian team’s request for faxed confirmation of meeting agendas and meeting decisions, which was at odds with the Western team’s desire for more informal communication.

The other major aspect of culture in software development is the cultures of customers who use the software produced. Literature in this area discusses the variety of cultures using software as a challenge to overcome, especially in the area of user interfaces. Greenwood points out the differing requirements by culture of presentation of date, time, numbers and other visual components⁶³. These cultural differences require sensitivity in preparing user facing software applications, but do not present as fundamental a challenge to overcome as adapting software to meet different functional requirements by culture. Greenwood alludes to the future of software where group collaboration will be different across cultures and thus software to facilitate this must include those differences. For example, he notes that business decision making in Japan is much more heavily based in group harmony, and thus any software which facilitates group

decision making must be designed differently to meet the Japanese need. Kersten et. al. extend this notion to discuss the “culture-dependent core” of software⁶⁴. They advocate for an attempt to identify and separate the culture-dependent modules of software from the core modules. One example given is that of the shopping cart, which is a fundamental part of the architecture for online merchant transactions, but is a notion that may not be familiar to cultures which do not shop in large supermarkets or department stores. The HomeNetToo project tested this concept, and confirmed that culturally-adapted interfaces were much more successful in conveying information to low-income African Americans in mid-Western United States urban areas⁶⁵. In the global marketplace, the audience for software may not always consist of only one culture, and so another important factor discussed in the literature is being able to adapt on the fly to different cultural audiences. A study at Michigan Technological University used corporate web pages from Lotus Development Corporation and Adobe to identify “cultural markers” which were design elements of the web pages such as color, text density and language that could be rendered in different ways after the user divulged their culture of origin⁶⁶. The teams discussed in this study, have the potential to take these challenges and move the discussion forward to the point where cultural flexibility in software teams brings functional flexibility which is valuable for a wider audience than simply the various cultural users.

Methodologies

In order to understand the broader context of global knowledge-based teams, experts from industry, academia and government were invited to participate in a seminar. The lessons learned from this context development were used to guide decisions on which forms of data to collect from the teams. The exact methodology used is described below.

Context Development

In understanding the nature of knowledge sharing across geographic borders it was first necessary to understand the broad context of offshoring of knowledge based services. The insights collected from this process were used to drive the methodologies for the case study portion of this project, by applying these insights directly to the domain of team-based software engineering. In the section below, we first discuss the seminar and the types of results we sought to obtain; then, in the results section of this report, we discuss how these specific results were used to drive the data gathering decisions made in the case study.

A seminar was conducted at the Massachusetts Institute of Technology, organized by the author along with Professors Lester Thurow and Amar Gupta of the Sloan School of Management. The goal of the seminar was to develop a broad context for the environment in which knowledge-based global work processes exist and the various factors which influence them. Thus, the seminar included experts from a variety of industries, not just software. Experts were chosen along three axes:

- **Role:** Consultant, Small-Medium Enterprise, Large Enterprise, Government, Analyst, Labor
- **Industry:** Software, Automotive, Health Care, Business Processes, General (non-specific)
- **Geography:** India, China, Russia, United States

The seminar was conducted in weekly meetings with three experts at each session. Experts were given a research prospectus on the issues we intended to understand regarding

knowledge-based offshoring, and then asked to give a 30 minute presentation on their particular perspective on knowledge-based offshore outsourcing. Students from various departments at MIT – primarily MBA students – attended the seminar and asked questions, many of which led to the results reported in this study. A table of experts who attended the seminar is provided here:

Expert	Organization Role	Industry	Geography
Bob Suh, Managing Partner, Accenture	Consultant	General	All
V. Srinivasan, CEO, ICICI InfoTech	Consultant	General	India
S. Khalsa, VP Sales, ICICI One Source	Consultant	General	India
Sanjay Saini, Dir. of Radiology, MassGeneral Hospital	Large Enterprise	Health Care	India
S. D. Shibulal, Co-founder, Infosys	Consultant	Software	India
Robert Reich, former U.S. Sec’y of Labor	Government	N / A	United States
Raj Malhotra, CEO, Spryance	SME	Health Care	India
John Ellis, President, Optics for Hire	SME	Mech. Eng.	Russia
Stephen Baxter, Sr. V.P., ERG	Consultant	General	China
John McCarthy, Forrester Group	Analyst	N / A	N / A
John Plansky, Chief Exec, Security, WIPRO	Consultant	Software	India
Renee Fry, Chief of Staff for Economic Development, Massachusetts Governor	Government	N / A	Massachusetts
Gio Wiederhold, research with IRS	Analyst	N / A	N / A
Marv Adams, CIO, Ford	Large Enterprise	Automotive	India
Joe Saliba, President, US and AP, CGI	Consultant	General	Canada / All
Tim Costello, Coordinator, North American Alliance for Fair Employment	Labor	N / A	United States

Figure 2. Experts on knowledge-based offshoring who attended the seminar.

Research Subjects

Two software teams at IBM were chosen for this study, based on the desire to control for as many factors as possible while testing the difference in geographic and temporal distribution as a treatment. For this study, the focus will be on the software developers on each of the two teams, and the technical, social and organizational factors which affect knowledge sharing. The individual research subjects will not be identified, as per agreement with the sponsoring organization, however individual identities will be distinguished to provide an accurate representation of the data related to each team member. In identifying the teams, consideration

was made of the many aspects of the software business in which IBM is involved – new product development, maintenance development and software services. The teams being studied were chosen because they produce yearly releases of new products, but also support releases in the market from previous years and in certain cases provide specialized services for particular large customers. Thus, they incorporate knowledge sharing requirements from all domains. In the future research section of this report, ideas for broader studies encompassing a wider variety of teams are proposed.

The controls for this study involve the structure, size, project lifecycle, technical domain, management structure and processes of the two teams. Both teams are formed of developers ranging from recently hired college graduates to experienced veterans of the software industry. Both teams are comprised of approximately 8 developers – team size fluctuated during the one year study – and have a tightly integrated small quality engineering team in the U.S., along with substantial testing done by a geographically separated team. Both teams operate on the same one year release cycle, with periodic maintenance releases in between the major annual releases. They are both managed by the same development manager, who is the direct line manager for all developers interviewed in this project and is responsible for the software development effort of both products. The products developed by each team are at a similar stage in the product lifecycle, having been introduced into market approximately five years ago. The technical domain of both products is persistent collaboration, with specific focus on content management. In addition to having the same development manager, the two teams also have the same project manager and thus follow very similar processes in terms of meetings, documentation and decision making.

With the above controls in place, the treatment in this study is the difference in team distribution of the two teams being studied. One of the teams, known in this report as the Collocated Team, is comprised of developers all co-located along the same hallway of an office building in the United States. One developer on the team did move to another location and continue to work with the team, and developers do occasionally work from home, but the primary work location for all team members is the same. The other team, known as the Distributed Team, is entirely geographically separated. The four developers in the United States are each located in a different location, and the other four developers are located in India. This difference between team location provides an opportunity to assess the impact of geographic distribution on the specific differences in knowledge sharing as exhibited by the technical, social and organizational systems in place within both teams.

The outcomes of knowledge sharing within the teams will be used to assess the impact of the team distribution. The primary goal of each software team is to deliver a new product release as well as resolve problems in the existing product, documented by Software Problem Reports (SPRs). These two outcomes can be measured in part by the performance to schedule of the teams, and the time to resolution of the SPRs – both of these outcomes will be tracked. However, since there are many factors which can influence these outcomes, a more direct outcome to study is the frequency and quality of communication among the team members, judged both quantitatively and qualitatively. The assumption here is that more knowledge sharing is better, and that higher quality knowledge sharing – i.e., knowledge directly useful in performing work tasks – is better for the teams. By identifying the ways in which this knowledge sharing takes place, and tracking the impact of this knowledge sharing on the work being performed, it is

possible to develop proposals which will leverage aspects of both teams' social, technical and organizational structure to make improvements.

Finally, it is important to note that the author of this study is a member of the Collocated Team. The impact of this participant-observer method of research on the outcome of this report will be discussed in a later section.

The teams were presented with a Research Prospectus which described the goals of the study being done. This is attached here as an appendix.

Quantitative Data

Quantitative data from the two teams described above was collected over a one year period in the year 2004. The development teams' main project deliverable is on a one year timeframe, so this period should cover every major point in the project lifecycle. Within this year, the teams also spend a considerable amount of time on short-term tasks such as attending to customer deployment issues and fixing bugs for maintenance releases – thus, the one year timeframe also provides an opportunity to gain insight on knowledge sharing for all scopes and varieties of tasks.

Within this one year timeframe, data was gathered from the sources listed below. These sources were selected to provide insight into the knowledge sharing from technical, organizational, social and process dimensions. Statistical correlations were drawn between various quantitative factors to propose links between these different dimensions.

Personal interviews

Hour long personal interviews were conducted with each of the developers on each team. While the focus of these interviews was primarily to gain qualitative insight, certain quantitative questions were asked to provide a general idea of the developers' own views of their knowledge

sharing requirements. The pieces of data collected are as follows, with explanations provided as needed:

- Informal Interactions per Week – informal interactions are defined as interactions which do not commence with an intention of discussing business.
- Main Developers Interacted With Informally
- Main Developers Interacted With Formally
- Time Spent Informally - In Person
- Time Spent Informally – Instant Messaging
- Time Spent Informally - Phone
- Tactical Decisions Made Informally – tactical decisions are defined as decisions minor in scope, with minimal knowledge sharing requirements and minimal impact on other developers' work.
- Strategic Decisions Made Informally – strategic decisions are defined as decisions major in scope, with significant knowledge sharing requirements and long-term impact on other developers' work.
- Strategic Decisions Speeded Up Informally
- Tactical Decisions Speeded Up Informally

Software Problem Reports

Each development team keeps track of fixes requested or made to the code base via Software Problem Reports (SPRs). The SPR contains information on the problem being reported, as well as the history of knowledge provided by various developers in resolving the issue and information regarding the actual fix to the issue. SPRs are stored in a central database for each team. For this study, a software tool was written to perform the data collection. This tool

analyzed SPRs fixed over the one year period of study, and collected the data described below, for each developer, on a weekly basis:

- **Main Developers Overlapped With** – For a given SPR’s primary developer, a listing of the other developers who provided input into the SPR.
- **Average Delay Between Developer Inputs** – For a given SPR, the average time between one developer’s input and another developer’s input.
- **Ratio of Collaborative to Individual SPRs** – A collaborative SPR is defined as one which includes input from more than one developer.
- **Average Time to Resolution** – The average time it takes for an SPR to move from being approved by the management team to be fixed to actually being logged as fixed.

Weekly Meetings over 1 year – coding system for tasks, status

The weekly meetings of each team were analyzed to provide insight into the formal task allocation and knowledge sharing on a group-wide basis for each team. The Collocated Team held one team-wide meeting per week, while the Distributed Team held one weekly meeting for the US based team members and one weekly coordination meeting between the development leads from the US and India. Each of these three weekly meetings was analyzed for the entire year.

Manual review of 3 meetings per week (1 for co-located team, 1 for US team of US-IN, 1 for US-IN joint session) was done to collect the following data, with respect to each developer:

- **Main Developers Interacted With**
- **Tactical Tasks Assigned**
- **Strategic Tasks Assigned**
- **Tactical Status Requests**

- Strategic Status Requests
- Developer Input Requested
- Developer to Developer Information Requests

Source Code Control System

Each of the two teams uses a source control system to log the modifications made to each element of the source code for the team's product. The source control system stores the date, time, developer making the change, and a comment regarding the particular change. The comments often site particular SPRs if there was an SPR which drove the particular change to be made.

The goal of collecting data from the source control system is that it provides a clear depiction of the technical system, to complement the social and organizational systems described by the other forms of data being collected. The data collected provides a representation of the technical dependencies between developers on the teams, and the rate of technical collaboration within the teams. The following data was collected, with respect to each developer, on a weekly basis:

- Main Developers Interacted With – In modules for which multiple developers check in code, the rate of shared check-ins with each of the other developers will be tabulated.
- Delay Between Check-Ins – the average time difference between modifications to a particular module.
- Reciprocal Check-Ins – the rate of check-ins for which one developer performs a check-in which is followed in time by another developer performing a check-in to the same module.

- If SPR cited, Avg. Developer Input on SPR – if the comments in a source control check-in refer to an SPR, the SPR will be consulted to determine the number of updates posted to the SPR.
- Avg. Time Since Last Check-In – this provides an idea of the amount of code which is actively being modified.
- Average Modules Checked In Per Build – Periodically, the source code is built into an executable version of the product. The frequency of this ranges from once or twice daily in the testing and fixing stages of the project lifecycle, to once every week in the design and implementation stages of the project lifecycle.

Group Email Exchanges

A software tool was written to analyze electronic mail (e-mail) sent to all members of the team. A “thread” refers to the entire set of messages written in response to an initial electronic broadcast or request for information. This data provided some insight into the use of broadcast messages to share knowledge on the teams. The following data was collected, with respect to each developer, on a weekly basis:

- Main Developers Interacted With
- Threads Contributed To
- Average Delay Between Responses For Initiated Threads
- Number of Threads Initiated
- Average Length of Initiated Threads
- Average Number of Developers Per Thread Initiated

Qualitative Data

Personal Interviews before data analysis

Personal interviews with each developer were conducted before the data analysis was performed, to gain qualitative insight into the various facets of knowledge sharing on each of the teams. The interviews were each one hour in length, and used the following set of questions as a guide. In cases where a certain line of questioning was leading to deep and useful insights, these questions were focused on and not all of the other questions were asked:

- What is the frequency and nature of interactions between team members?
 - Both co-located and geographically dispersed
 - How are methods of interaction chosen: e-mail, IM, phone
 - One on one, or group?
 - Formal or informal?
 - Role-based or not?

- What is the past history of this team?
 - Recently introduced geographical dispersion?
 - How did interaction change from before to after?

- What is the organizational structure of this team?
 - Which tasks are geography based and which are project/team based?
 - Are there differences between reporting structure vs. operational structure?

- Is most work activity-based or team-based?
 - Are ad-hoc “teams” formed around particular activities?

- How are knowledge handoffs done?
 - Over different stages of project lifecycle
 - During the introductory, steady state, and transition periods
 - With regard to creative vs routine work
 - With regard to tactical vs strategic work
 - Can you provide examples of any of the above

- How does the team structure affect the knowledge handoffs?
 - How are social relationships associated with technical dependencies?.
 - Can you compare this experience to past teams?
 - Do you have suggestions on how to use your team’s structure to improve on the issues raised above?
 - Are there cultural differences which play a role?
 - What is the impact of time zones – positive and negative?
 - Are there any other pros or cons to your team’s structure?

- Are there any other thoughts you’d like to share?

Personal interviews after data analysis

After the data analysis was done, a short follow up interview was done with the developers to inform them of the results shown from the data analysis and ask if they had any feedback to provide based on awareness of the data results. These interviews provide an opportunity to extract more insight from the developers, and also gauge the level of awareness the developers have with the various results found from the data.

Participant – Observer model of research

As mentioned earlier, the author of this study is also a participant on the Collocated Team. This participant-observer model of research is one which can yield stronger results, provided there is an awareness of the drawbacks, as discussed in the prior literature related to this form of research. In this case, the benefits of having a participant act as the observer involve a deep understanding of the work processes, personalities and projects being undertaken by both teams which allows the observer to ask the in-depth questions that Myers notes only a participant-observer can ask. The participant has already developed a level of trust with the teams similar to that described by Elliott and Scacchi, and thus is able to gain access to richer data sets and possibly extract richer information through the interviews to meet the requirements posed by Seaman's model. There is also a general set of understanding related to the company and industry which aide in making decisions on which types of data to collect, and how best to access these forms of data. Finally, the participant has a greater incentive to gain a better understanding of the group dynamics, and has the opportunity to implement the proposals resulting from the study. It is important to note that for all of these decisions which are ostensibly made easier by the participant, there are likely to be biases based on the personal experiences of the researcher, as suggested by Poltrock and Grudin. As a participant, certain aspects of the knowledge sharing within the teams may have seemed more important, whereas an independent researcher may have

formed questions which were not biased toward one aspect over another. There is also the possibility of hesitation in reporting negative results, as a participant on the team. With these disclaimers being made, the important issue is that the study was conducted with an awareness of the potential drawbacks and an attempt to mitigate any downsides of being a participant while taking advantage of any benefits.

Stakeholder Analysis From Expert Seminar

The experts brought into the seminar provided a set of external political, economic, cultural, and social factors that would impact the efficacy of the 24-Hour Knowledge Factory approach for a particular software development endeavor. These factors have been compiled into a system diagram which shows the relationships between the various factors. The diagram here is not exhaustive – instead, it is intended to provide context on which to build questions for the in-depth analysis of the software teams. Thus, the diagram is followed by a short description of some of the key components used in building the case study.

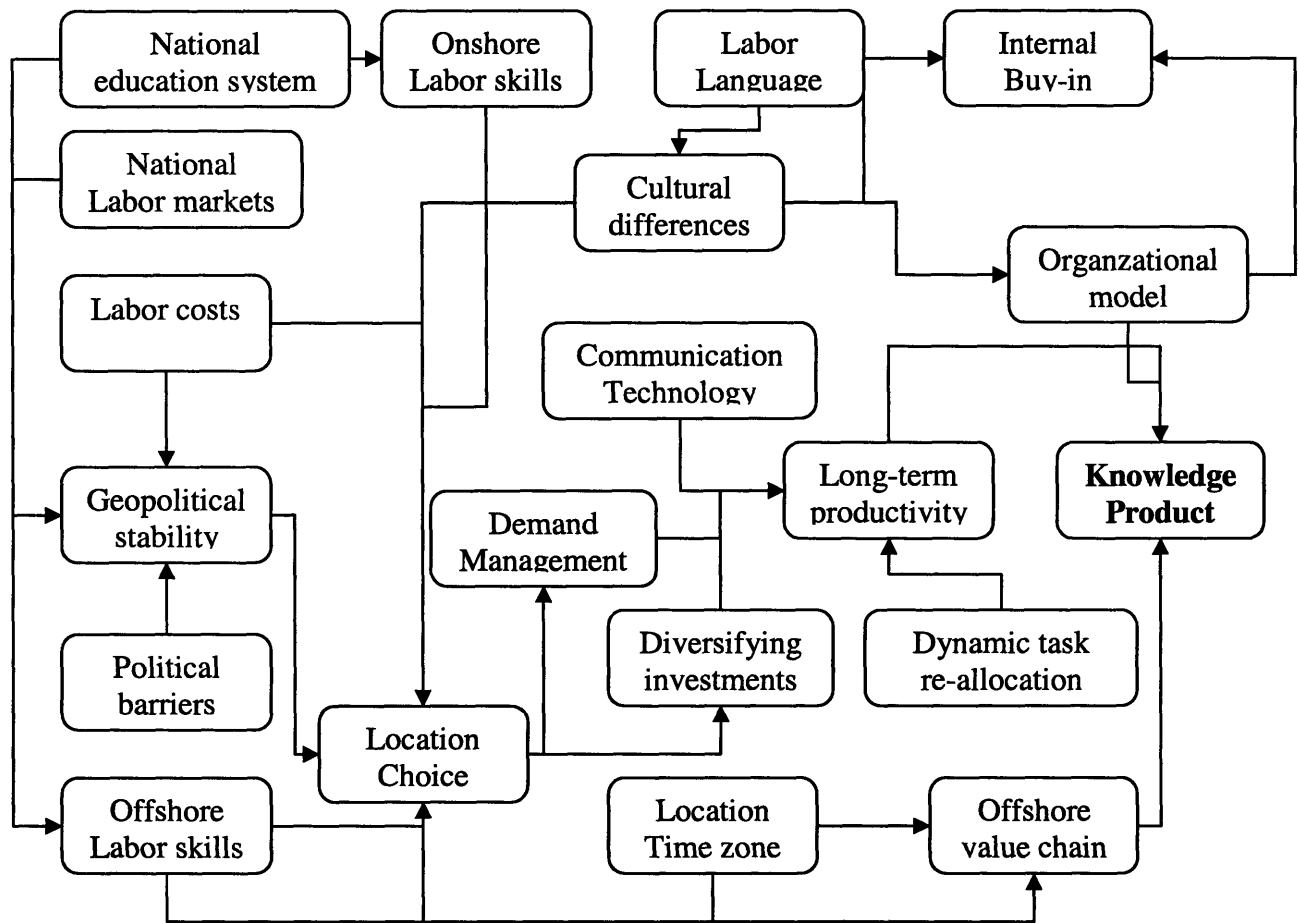


Figure 3. Systems model for factors influencing global knowledge-based work.

The diagram highlights the complexity of global software development, and the underlying interdependencies of the many factors involved. All short and long term factors must be assessed not only in their individual context, but in the context of the larger system.

Many firms who have set up an offshoring process have encountered similar strategic issues that need to be reconsidered when moving to a truly global development process. The seminar revealed that these issues emerge in firms of various sizes – from startups and small enterprises to medium and large sized multinational firms – and in various global delivery models – from third-party vendor relationships to captive centers.

Demand Management

A 24x7 development model allows for much greater management of customer demand. This model will promote a faster time to market for products, and will also allow the firm to adapt quickly to changing market conditions; this is because of the lower labor costs, the greater flexibility to reallocate and reassign resources, and the ability to provide customers with access to skills that they may not already have⁶⁷. A good example of the improved demand management provided by the offshore model is in the area of radiology, where offshore radiologists can read X-rays overnight and provide much better care, especially in an industry where the labor supply in the United States is limited⁶⁸. As offshore knowledge workers gain experience and move up the learning curve, their experience interacting with customers will allow them to broaden the scope in which they serve customer demand and provide for 24 hour availability of high value resources⁶⁹. One example of this growing phenomenon is a company that employs home-based workers in India to perform medical transcription. As these workers move up the value chain, their home-based work environment continues to allow them to be readily accessible. Accordingly, these workers can work longer hours, as necessary, concurrently

with family obligations at home, thereby serving as ‘agile’ knowledge workers in the knowledge factory in real-time⁷⁰.

However, in all of these environments, a key consideration is the ability to manage customer expectations, and to prevent end-customers and end-users to assume that they will be achieving results far exceeding what the offshoring model or the hybrid model can provide⁷¹.

Long-term Productivity

The use of offshoring or hybrid work models can provide access to higher skilled labor for tasks that previously only were done by lower skilled workers. For example, highly skilled radiologists in the United States are much less likely to prefer reading X-ray results while in India, a highly skilled radiologist will see employment by a U.S. hospital as a high-value position regardless of the task¹². When moving toward a 24 hour knowledge factory model, factors such as ability to grow in size, quality management, and the added communication and coordination costs must be incorporated into the calculation of the improved productivity^{11, 14, 72}.

The key here is to “transform, not transfer” the work¹³. The 24 hour knowledge factory is a significantly different paradigm for knowledge-based work, and it is not sufficient to simply send the work offshore when additional productivity gains may be realized by transforming the tasks into ones that are more appropriate for the global model. As these tasks become transformed, jobs may be lost or redefined, and the important point in managing employees who are affected by these changes is to ensure that employees can continue to be productive and add value. One appropriate analogy is the “law of the horse” that relates to the impact on horse carriage manufacturers just after the automobile was invented. Initially, the workers building horse-driven carriages saw their jobs vanishing; however, the overall impact on the job market

was positive based on the introduction of the automobile; the advent of the disruptive technology forced the horse carriage manufacturers to adapt¹⁵.

Similarly, the advent of the 24-hour Knowledge Factory paradigm will require pioneers adopting this paradigm to devote significant time in the beginning of the process in requirements gathering, organizing stakeholder workshops and setting up communication norms, in order to realize major productivity benefits in the long-run⁷³.

Integrated Value Chain

The application of the 24-hour Knowledge Factory paradigm explicitly implies partial offshoring. While the initial effects of such offshoring will be an increase in productivity, the offshore workers will gradually move up the value-chain and provide a great deal of higher-value services. This movement up the value chain must be acknowledged from the beginning, and planned for in terms of wage increases, infrastructure improvements and task redistribution. One option is to employ offshore sites at various locations, and as various locations move up the learning curve at various speeds, redistribute tasks appropriately between sites. Statistics show that the initial investment in offshoring is lower on the value chain. According to Accenture, for IT offshoring, 51.9% is in IT services such as maintaining computer networks, 36.7% is in solutions development such as building websites and 11.4% is in leadership and managing projects¹³. The general progression can be characterized as a movement from efficiency to innovation to growth, with production moving from commodities to services to solutions, as vendors begin to do similar work for multiple customers^{11,13}. The movement up the value chain is not reserved simply for the offshore workers, as in the example of radiology, US doctors can move to higher value tasks if X-Ray reading is done offshore¹². Different geopolitical locations possess their own characteristics that impact their current and future place on the value chain.

For example, China has lagged behind India in knowledge-based offshoring because of a lack of English speaking citizens, but as English becomes a more common language in China, the vast size of the labor pool will allow it to rapidly move up the value chain⁷⁴. In Russia, many highly skilled PhD scientists and engineers saw a dramatic drop in high skill tasks after the Cold War; these domain experts are now making a dramatic transition into high-value services such as optics design at a much lower cost to outsourcing firms¹⁵. The hiring process is a major factor in moving firms up the value chain, as a constant reevaluation is required of whether the foreign employees are indeed the highest skilled in their area^{16,75}.

Organizational Models

In order to maintain flexibility, one needs dynamic models that can evolve as market conditions change and learning curves impact skill levels. In choosing a model, it is important to judge the complexity of the work required and determine the right locations for each particular skill required¹³. This may lead to a model where the same function or skill is located in multiple geographic locations; this may involve higher management overhead but may lead to greater returns especially when taken in the context of the 24-hour knowledge factory^{16,19}. Two matrices upon which the organizational models can be judged are coordination versus effort, and complexity versus project size¹¹. These axes should be frequently revisited, especially since offshore and hybrid engagements can commonly move from project-based engagements to long-term contracts that may require a different model¹⁵.

Other Decision Factors

While cost is the primary driver for firms to outsource, the 24 hour knowledge factory embodies some of the secondary drivers that may exist for expanding into an offshore model. Such a model can lead to expanded opportunities for demand management, as described above,

but can also lead to quality improvements, improved access to expertise, and flexible staffing^{16,19}. The decision to offshore can not be based simply on cost because the correlation of the labor to the tasks is a very important consideration that should be revisited as tasks and workforces evolve^{11,14}. In picking the right foreign partners for the 24-hour collaborative endeavor, the final decision should be based on a thorough “kicking the tires” approach of visiting and testing the offshore organizations and ensuring domain expertise, with an understanding that the lowest cost vendor may turn out to be the most expensive in the long-run if the other decision factors do not apply¹⁷.

Common Barriers within Firm

Significant barriers to employing the 24-hour knowledge factory concept exist within typical firms; several of them need to be addressed as part of the initial decision process rather than as a corrective measure at a subsequent stage. Internal resistance, especially due to a loss of control, may hinder a proposed project¹¹. Furthermore, cultural, language and trust issues need to be approached in an upfront manner, recognizing the impact with respect to the interaction required between knowledge workers in the 24 hour knowledge factory^{11,16,17,18}. Even if the desire exists at all levels to pursue the globally collaborative engagement, the firm should plan on process changes such as longer project planning cycles, more explicit definition of requirements and communication methods and the effects of ill-informed hiring decisions¹⁹.

Location Choice

Inter-twined with the decisions related to whether to employ offshore resources, which model to pursue, and how to mitigate barriers, is the ultimate decision of which location or locations to pursue. Certain countries have identifying features which have made them popular locations for providing offshoring resources: India has a highly educated, English-speaking, IT

skill base; China has an enormous labor pool¹⁸; and Russia has a legacy of USSR investments in science and a solid brand that has not been exploited to the level of India^{15,76}. Regardless of the locations being considered, factors to consider include the geopolitical stability of the country, the investment in education, labor and skill set of citizens, and the business environment in the country, including levels of corruption and ease of setting up businesses^{11,18,20}.

Linking the Seminar to the Data Gathering

A number of the issues raised by the stakeholders in this study were used to drive decisions around which pieces of data to focus on in the case study. The potential for improved demand management indicates that customer fixes may be handled in a faster manner with around the clock effort, and this issue was discussed in interviews with team members from both teams. Long-term productivity gains from outsourcing suggest that it is important to look at how tasks are transformed when moving into the offshore model, and this served as a means of explanation for the variance seen in the data. The integrated value chain discussed in the seminar was used to help explain how different work locations can offer different benefits based on their own interests and skills, and this was definitely seen in the distinction between the two teams studied. Many of the common barriers to offshore outsourcing were discussed in the individual interviews conducted with developers, and in observing discrepancies with the data, the common barriers were used as a means for potential explanation. Stakeholders in the seminar discussed other decision factors besides cost which led to a decision to distribute work globally, and this was directly relevant to the team involved in the case study because the decision had already been made and many short-term decisions about which tasks to do in which location depended on the various other decision factors raised by the stakeholders. Finally, the organizational models shared by different stakeholders provided context for suggesting means of

improvement to the developers in the case study and soliciting their reaction. The stakeholder seminar was indeed quite useful in deciding which questions were most appropriate to ask, analyzing and explaining the data and in providing a general context which was much broader than that which the individual developers working in the environment could provide.

Core Findings

Structural Determinants of Coordinated Development

This section discusses the findings related to the specific forms of data which were collected. These data provide insight into the differing ways in which the two teams at IBM use the same processes and technologies to accomplish their goals. The variance in the 52 weeks of data suggests that there are certain structural determinants which can be used to distinguish distributed and co-located teams, but that these structural determinants do not define the destiny of the team.

The two software development teams studied were virtually identical in all structural respects. Each team had seven core developers, of similarly varying experience and responsibility. Both teams are managed by the same development manager, who is responsible for guiding the technical direction of the product and ensuring that the team has the resources they need to complete their tasks – the developers who were interviewed in this study all report to this same development manager. Both teams also have the same project manager, who is responsible for ensuring that processes and schedules are set and followed, and for keeping track of the team's progress. It was especially useful to have the same project manager in studying the team meeting minutes, because each team's meetings ran in a similar format to the other and were also recorded in the same format, by the same individual. Both teams have the same project schedule, and ship their individual products as part of the same suite of software products. Each team delivers one major release every 12 months, and also delivers periodic fix packs to customers as requested.

The two teams were also as identical as possible in terms of the technical aspects of their work. Both products have been in market for a period of years and are considered mature but not

yet in maintenance mode. Both products are in the technical domain of content management – they are used by large enterprise customers to store versions of internal and external documents and collaborate with team members. It is important to note even though they are in the same domain, the products are complementary rather than competitive and serve differing markets. The two products each require the same skills and technology to build: C++ is used as the primary programming language for the server component of the product and each product includes a Java based application programming interface (API).

While controlling for all aspects described above and treating the geographic distribution as the key differentiating factor, the data presented in this section teases out the full range of contrasting aspects of each team's information sharing processes. All charts study the week to week output of the given data set for the year 2004. This format is used for consistency, however in some cases the aggregate data is more useful than the time series data in comparing the two teams.

Using Electronic Mail for Asynchronous Discussion

The data collected showed that the distributed team made much greater use of electronic mail (e-mail) as a forum for discussion. This usage peaked during the time periods following project deadlines and was relatively constant in the periods of steady work well before the milestone. The collocated team relied on e-mail as an announcement mechanism for broadcasting a message to the general set of developers, but relied on other means for back and forth discussion. The four charts below discuss aspects of these findings in greater detail.

The e-mails which were tabulated for each week of the year, and only those emails which were sent to the entire group were tabulated. Further study in this area could obtain access to

individual e-mail boxes and observe individual mail communications, but for the purposes of this study IBM only provided access to e-mails sent to the group as a whole.

Average # of Contributors Per Email Thread (each thread accounted for in the last week an email was sent)

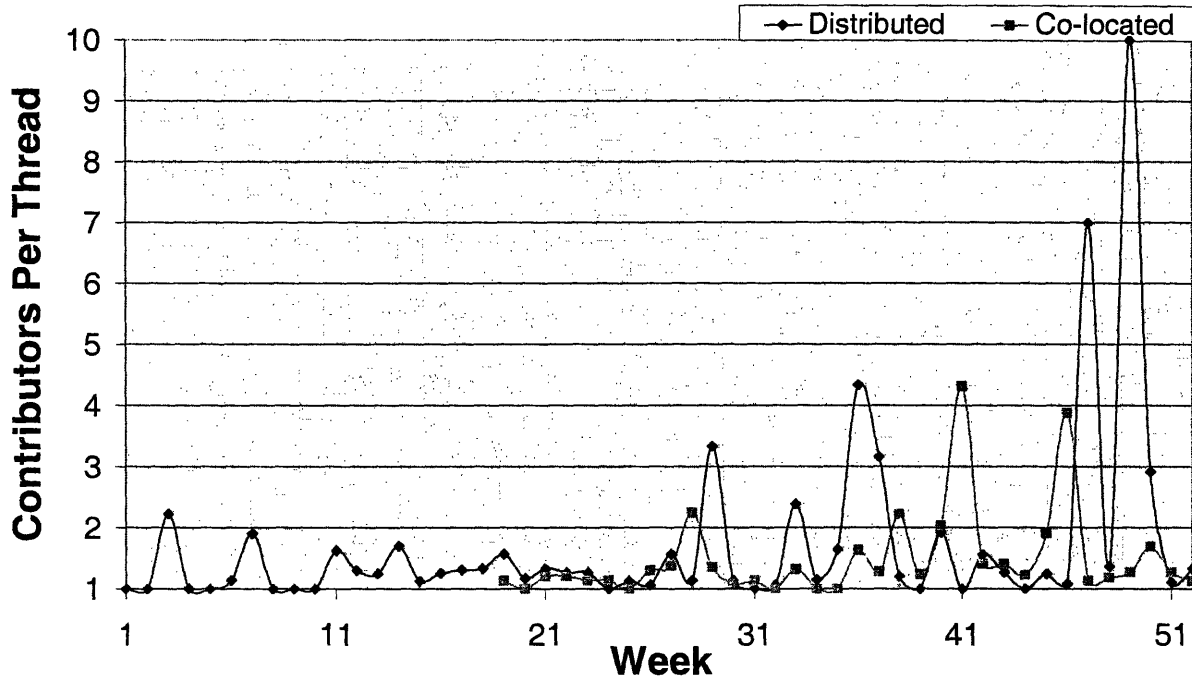


Figure 4. Average number of contributors per email thread.

The chart above displays the average number of contributors per e-mail thread. A thread is defined as an initial message, and the series of messages which are in response to either the initial message or other messages in the thread. This was calculated by grouping messages with the same core subject line together as belonging to the same thread. It is often the case that a reply to an email will contain a "Re:" or "Fwd:" preceding the subject, and this was accounted for in grouping threads. There were also a small number of cases where the sender of the response e-mail changed the text of the subject line, and these cases were not accounted for in this chart because it would require obtaining access to the content of the message to determine the preceding threads.

It is also important to note that the chart is displayed with the thread being accounted for in the last week for which an email was sent in the thread. Thus, the apparent high volume of threads in the last weeks of the year also incorporates thread activity from earlier in the year. Later charts in this series will address the raw number of emails and threads being sent during the year to provide a more accurate time series picture of this data. This chart is primarily useful for observing the differences in number of contributors to email threads on an aggregate basis.

The average number of contributors per thread for the distributed team was 1.73 and for the collocated team was 1.50. In the following charts, this data will be compared with data on the raw number of threads to show that in general the e-mail usage was much higher in the distributed team. This chart demonstrates that e-mail is used as a means of long lasting discussion on the distributed team, while it is primarily used for one message announcements on the collocated team. A developer on the collocated team stated that many of the one message announcements from the collocated team are announcements that a particular individual is heading out of the office, even for a period of just an hour. This demonstrates a significant difference in team culture on both teams. On the collocated team, face to face discussion is so important that team members feel the need to inform each other if they are going to be unavailable for a short period of time. On the distributed team, long discussions are done over e-mail and often last days because of the time zone differences. A developer on the distributed team cited one of the benefits of having discussions done over e-mail is that team members can take the time to think about their responses and often provide more detailed input into the discussion. The members of the distributed team also all stated that when discussions reach a significant length, they are moved to a discussion forum database where responses can be better tracked and archived. Distributed U.S. Developer 2 noticed a benefit to the resulting hybrid form

of discussion incorporating both synchronous and asynchronous communication. For example, when reviewing the design of a feature to be included in a product release, it was common for the U.S. portion of the team to hold a meeting with a presentation to discuss the design. Later that day, after the U.S. work day was complete, the Indian portion of the team would review the slides from the presentation, and provide feedback in an organized and written manner. Distributed U.S. Developer 2 noticed that both forms of feedback – immediate face to face and asynchronous written – were useful in the end product and neither would have been achieved if operating in the other framework. Distributed Indian Developer 2 stated that "one of the things I have found that helps when working in cross-cultural teams is to set ground rules and mechanisms for communications." The setting of ground rules was acknowledged by all team members as an important factor in both overcoming cultural differences and encouraging uses of the different technologies discussed in this study. The manager of both teams stated that the discussion forum database is common to both teams, however the distributed team is much more prone to using it. This is a likely result of the distributed team's familiarity with having to carry out discussions by typing their responses in e-mail format.

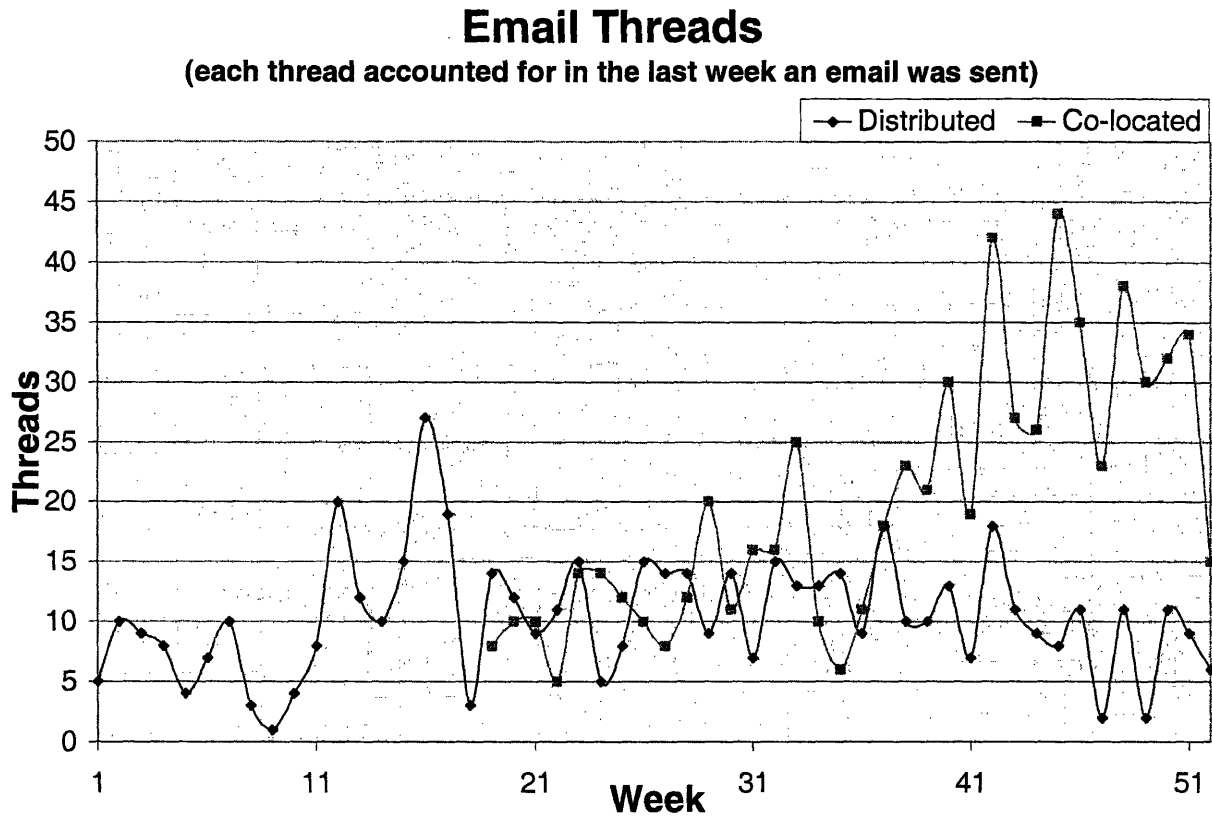


Figure 5. Email threads created by team members

This chart displays the raw number of email threads sent by each team – it is especially interesting because unlike the first chart, the volume for the collocated team is higher than the distributed team. Once again, the threads are accounted for in the last week for which an email was sent in the thread. The average number of threads for the distributed team was 10.42 and the average number of threads for the collocated team was 19.85. When taken in context with the first chart, these data confirm that the collocated team has more e-mail threads created, but many of these threads have just one contributor. The distributed team has a significantly less number of threads, but on each of these threads has a high degree of collaboration. This confirms the anecdotal evidence from the interviews that the nature of e-mail use and the nature of knowledge based discussions on both teams is profoundly different.

Another interesting aspect of this chart is each team's reaction to a project milestone. Week 41 was the "feature freeze" date for both teams. This is the date at which code for the features being delivered in the next release of the product needs to be provided by the developers and sent to the testing team. After this week, the number of threads in the distributed team goes down to an average of 8.91, while the number of threads in the collocated team goes up, to an average of 31.45. A developer on the collocated team reacted to this data by noting that many problems are found by the testing team when features done by different developers interact with each other, and e-mail discussion is used to quickly remedy the gaps in knowledge between developers who worked on different but interacting features. This suggests that the distributed team's higher volume of threaded discussion in the weeks preceding the milestone allowed them to more formally address the issues which may arise from individual developers' features interacting with each other.

Average Number of Emails Per Thread (each thread accounted for in the last week an email was sent)

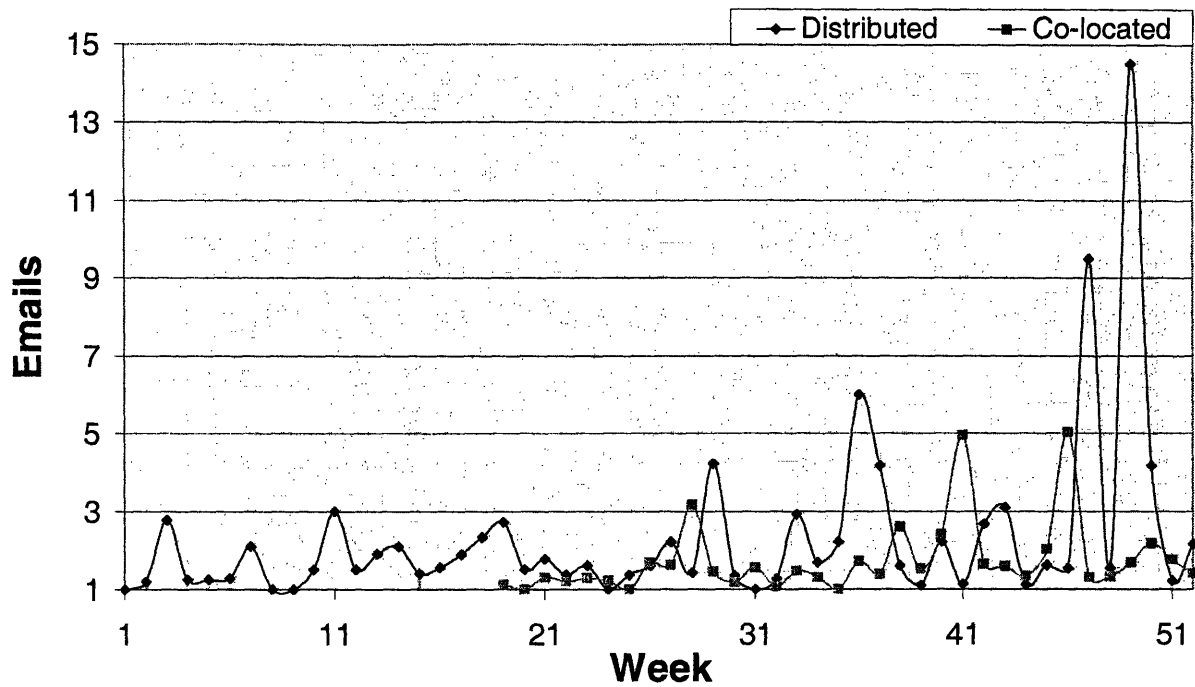


Figure 6. Average number of emails per thread.

This chart is a reinforcement of the first chart – it demonstrates that longer e-mail discussions are the norm in the distributed team. The first chart addressed the number of individuals contributing to each thread, and this chart addresses the number of emails sent per thread. The average number for the distributed team was 2.32 e-mails, while the average number for the collocated team was 1.75.

Email Messages Per Week

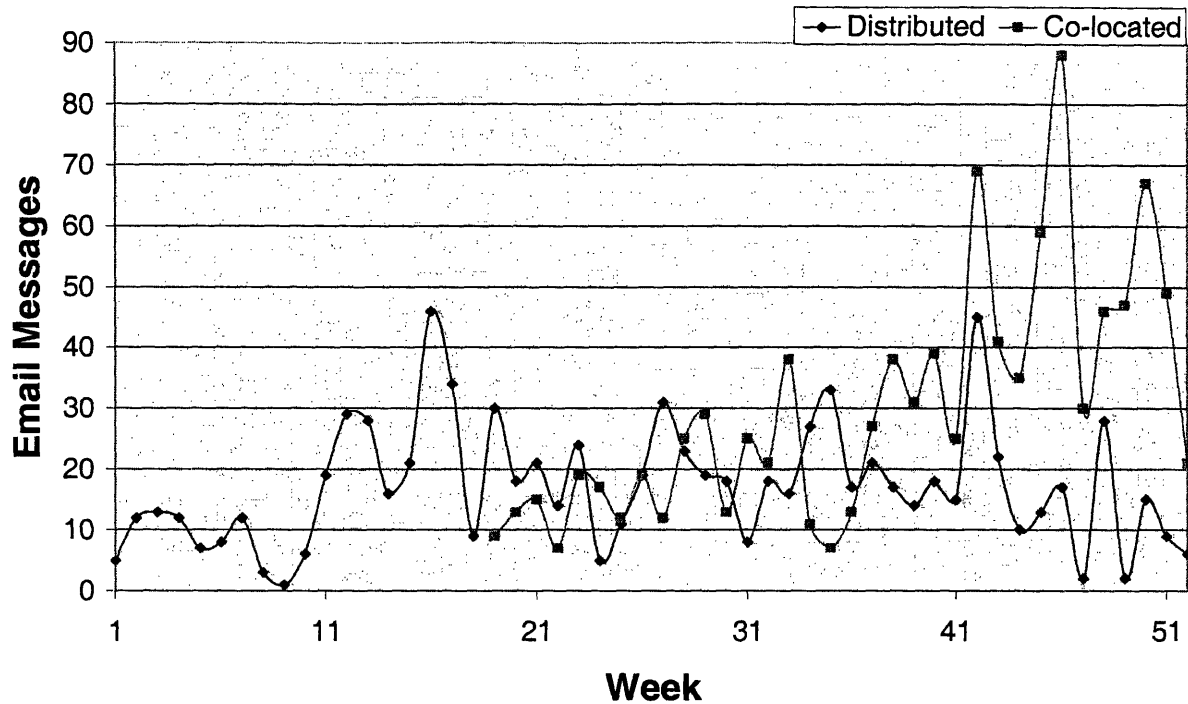


Figure 7. Email messages sent per week to all developers on each team.

This chart is a reinforcement of the second chart which showed e-mail threads per week. Once again in this chart, the email activity is greater in the collocated team, and the email activity increases significantly after the feature freeze project deadline. The average number of emails in the distributed team was 17.06 while the average for the collocated team was 29.91. As stated before, the increase in activity after the project milestone suggests that the collocated team found a greater use for information requesting and disseminating over email after the feature freeze passed – perhaps because the need for quick resolution was greater and the decisions were more tactical than strategic.

Technical Collaboration through Shared Source Code

Data from the source code for each team's product provided insight into the technical collaboration between the two teams. Each team reacted differently to project deadlines, with

the level of activity in the collocated team being more controlled before the feature freeze date but increasing afterwards. The collocated team was found to have a higher degree of collaboration with respect to specific code elements; while the distributed team kept the code they modified separate from each other.

Source Code Modifications

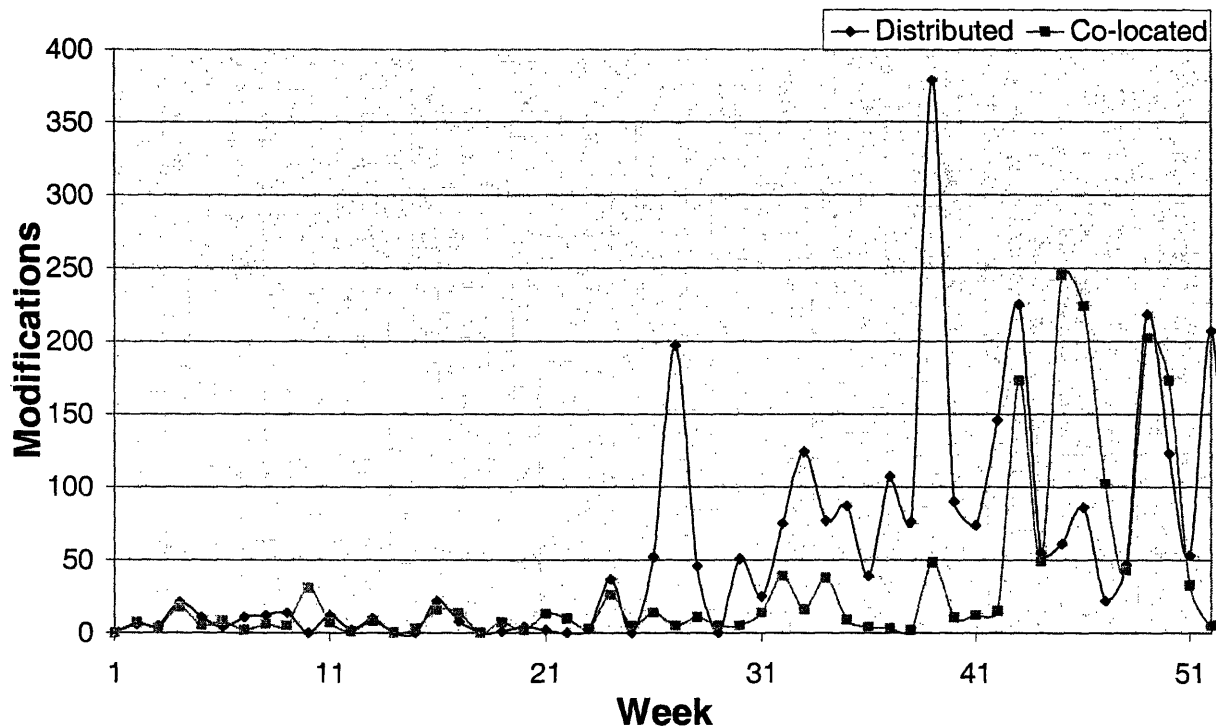


Figure 8. Number of source code modifications per week.

This chart displays the raw number of source code modifications per week for each team. For each product, the source code control system contains one “branch” for each version of the product which is in market or being released. For versions which are in market already, the branch is modified when a customer fix is requested and source code changes are needed. For the upcoming release, a branch is maintained and developers modify code in the branch when they are satisfied with a particular feature or fix that is meant to be included in the upcoming release. The modifications shown above refer to the number of code elements modified – it is

important to note that whether a modification involved extensive changes to the code element or a simple fix, for the purposes of this study they are logged as one modification. Further study should distinguish between major and minor changes, and between the versions of a product.

The weekly averages for different time periods of the project were calculated to provide a picture of how each team reacted to different parts of the project. In the period up to week 22, the average for the distributed team was 6.73 and the average for the collocated team was 7.68. This suggests that both teams handle the steady state before a deadline in the same manner. In the period from week 22 to week 41, as the feature freeze milestone was being reached, the average for the distributed team was 84.10 modifications and the average for the collocated team was 14.10. This suggests that the collocated team is able to handle the collaboration before a deadline in a steadier manner – the interviews with the collocated team speculated that this was due to questions being resolved face to face with individual developers consulting others before submitting a code modification. In the period from week 41 to week 52, after the feature freeze milestone occurred, the average for the distributed team was 100.45 and the average for the collocated team was 114.27. These two last time periods suggest that the distributed team did not uncover as many problems with each developer's code being submitted for the feature freeze problems. With the collocated team, activity rose after the feature freeze when code from individual developers was made to interact with code from the other developers. The next chart provides some insight into why this may have occurred.

Average # of Developers Per Code Element (each element accounted for in the last week of modification)

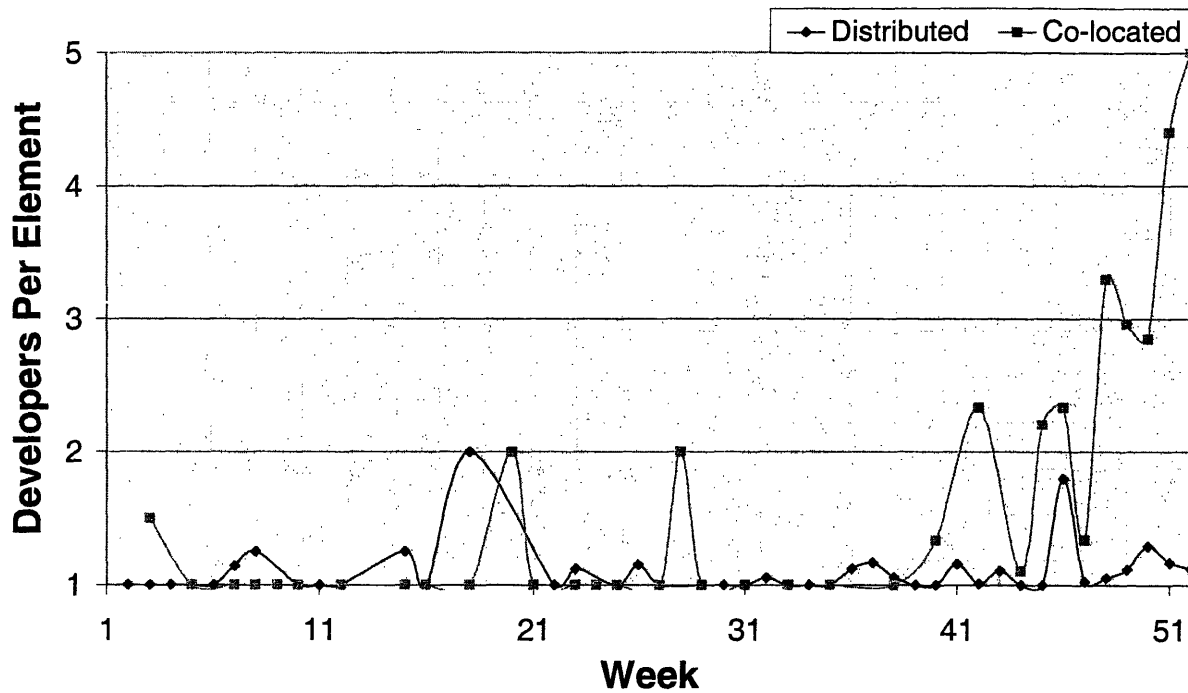


Figure 9. Average number of developers modifying individual source code elements.

This chart displays the average number of developers per code element, with each element accounted for in the last week that it was modified. The elements are only accounted for in one week each even though collaborative activity could have taken place over a period of weeks. Thus, it is important to note, similar to the data regarding email threads, that the time series in this data is not meant to provide an accurate picture of the collaborative activity from one time period to the other. Although the data could be further massaged, the purpose of this study was served by the presentation provided here.

These data demonstrate that the collocated team definitely has more overlaps in terms of developers working on the same code elements. The weekly average number of developers per code element for the distributed team is 1.10 and the collocated team's average is 1.63. Multiple team members on the distributed team cited the fact that they have drawn clear lines between

code elements and make an explicit attempt to only modify certain elements of the code. Distributed U.S. Developer 3 stated that "everyone's sort of off in their own little world" but did not cite this as a negative aspect of the team's productivity. The only downsides he identified were when one member of the team left for a vacation and knowledge from that area of the code was needed. Distributed Indian Developer 3 felt the technical separation was required in the beginning stages of the project: "you have to give time and space to people to work in their frames till both sides get slightly accustomed [to the other]." By contrast, while the collocated team does assign particular functional areas of the product to different developers, they will often reassign particular SPRs based on workload and feel comfortable with any developer modifying any part of the product. It is important to note that while the data do suggest more technical collaboration on the collocated team, there are code elements on the distributed team which have more than one developer. Thus, even when distributed, the software developers do reach out to others for help when certain threshold barriers for requiring higher levels of collaboration are reached. Further study can examine when these higher barriers are encountered and how technical teams can be redefined to accommodate these barriers.

Nature of Team Meetings

The team meetings held by the distributed team were found to be more tactical and task oriented than the collocated teams, further demonstrating that each team has adapted similar processes in different ways to their own geographic structure. The two team's meeting minutes were controlled by many factors which allowed data collection to proceed with confidence. All meetings on both teams were held by the same project manager, who kept detailed minutes of each meeting in the same format. The categories for the agenda changed over the year to fit the stage of the project schedule, but were generally found to be consistent between the teams.

Before reviewing the data, it is important to note that these data represent an accurate snapshot of the meeting minutes, which represent what the teams each decided was important to discuss and record in the meeting report format. These data do not represent a full account of the activities which occurred week to week.

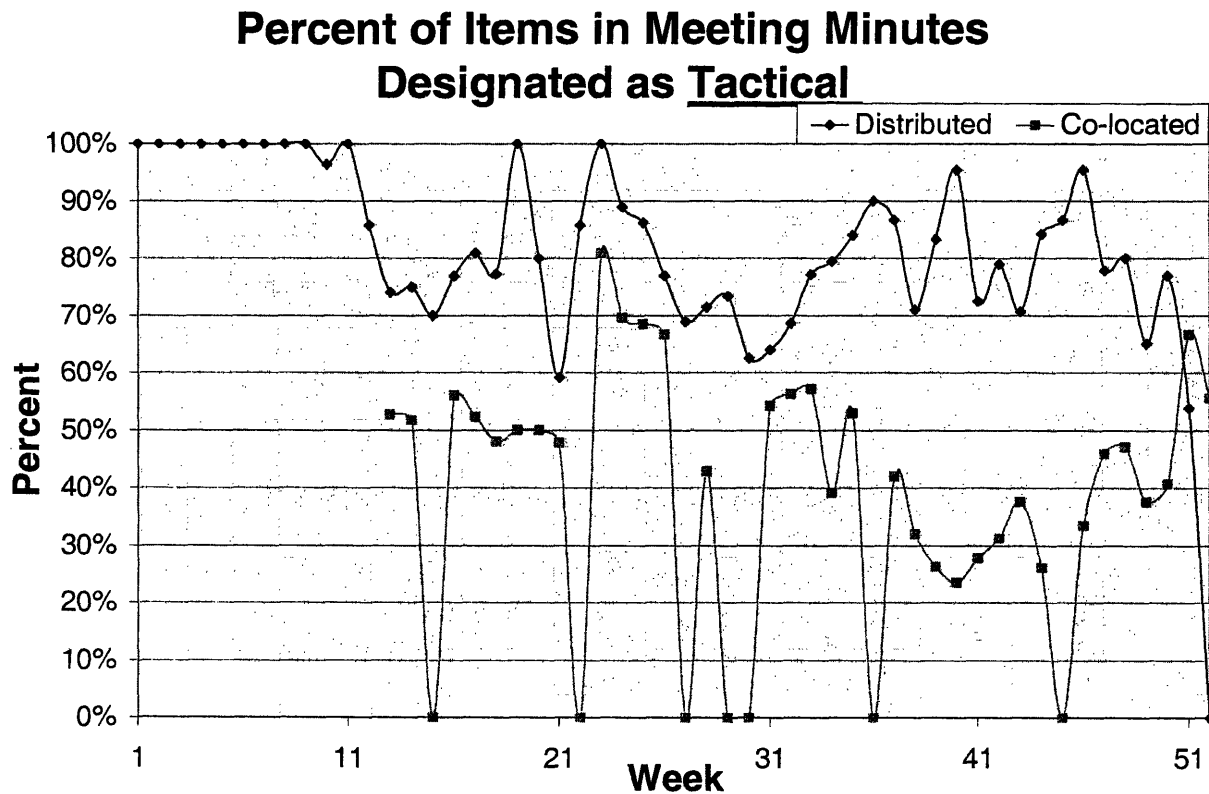


Figure 10. Percent of items in team meeting minutes designated as tactical (as opposed to strategic).

The data in this chart shows that the distributed team devoted a larger number of items in the meeting minutes to tactical issues. As discussed in the Methods section of this report, items were designated as tactical if they were short term in nature and all knowledge related to completion of the item was already acquired. The meeting minutes were analyzed by inspection, and all items in the meeting minutes were either designated as strategic or tactical, no other categories were used. Examples of tactical items found in both teams' meetings included issues

related to the “build” (a compilation of source code into an intermediate internal product release to be sent for testing), issues related to a particular SPR, or issues related to scheduling. Examples of strategic items found in both teams’ meetings included discussion of feature plans for the next release, discussion of major customer issues, and discussion of cross-team collaboration with other teams in the company. The average percent of tactical items found in the distributed team was 81.36% and the average number for the collocated team was 39.23%. This suggests that the collocated team finds ways of handling the tactical issues outside of the formal meeting structure, because opportunities for synchronous communication are available. Collocated Developer 4 stated that it is rare to wait for the meeting to discuss tactical items because instant messaging or face to face office visits are a regular part of the team culture. On the other hand, Distributed U.S. Developer 3 stated that meetings were the one change the team had to be together and he often compiled a list of action items he wanted to discuss in preparation for the meetings.

Percent of Items in Meeting Minutes Designated as Tactical or Strategic Task Assignments

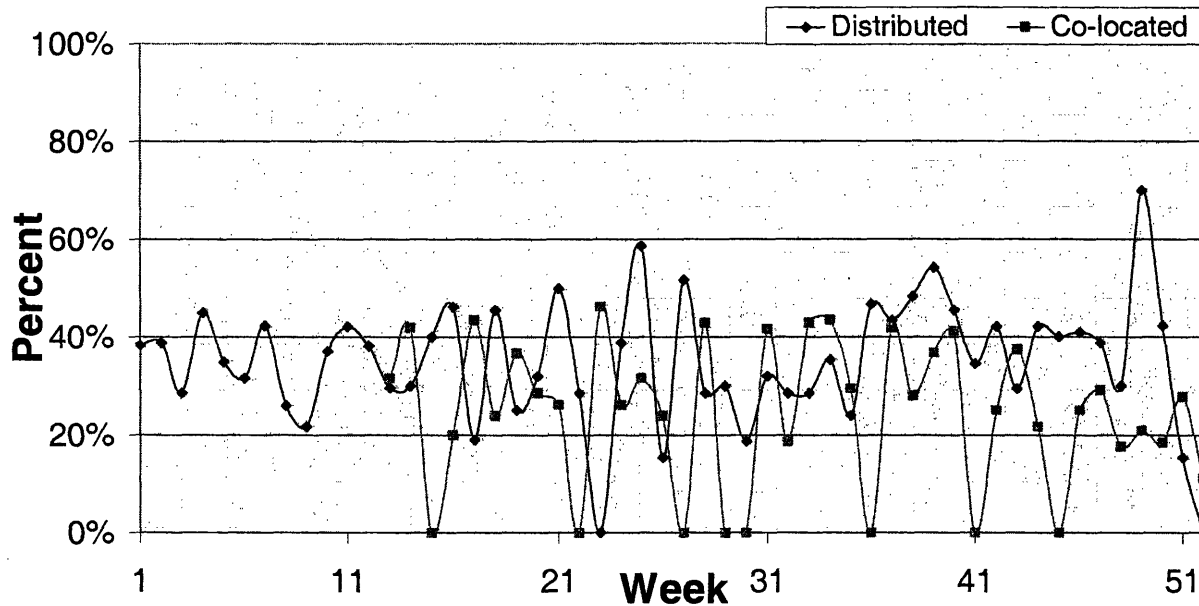


Figure 11. Percent of items in meeting minutes designated as task assignments (as opposed to status updates).

This chart demonstrates two interesting trends – the distributed team spends a larger percent of meeting items on task assignments but both teams exhibit an alternating behavior of switching between task focused and status focused meetings. To compile this chart, all items in the meeting minutes were designated as either task assignments or status requests. This was done in parallel with the tactical vs. strategic designation, so all meeting items received two designations and could be tactical task assignments, tactical status requests, and so forth. The average percent of items which were task assignments for the distributed team was 35.08% and for the collocated team was 24.53%. The alternating nature of the curve is seen from inspecting the chart – on a week to week basis the level of task assignments alternates from a high point to a low point, with each high and low range staying relatively constant over the course of the year.

This variance could represent either an action/re-action mode where tasks are assigned and then status is requested, or it could represent instability in the team, or it could just be an accidental result. The interviews with team members did not yield any specific explanation for this variance, but it is important to acknowledge and identify for future research.

Using Technology to Update Work Item Status

Data from the Software Problem Report database was useful in demonstrating how technology is used to update work item status. SPRs represent the core work items for these software development teams, outside of feature level work. When any work is required on the source code – either a bug found by the testing team, an enhancement requested by a customer, or a feature – an SPR is logged and is used to track the status of the work. Under a new system which both teams are moving towards in 2005, any source code modifications are required to have an SPR which documents the reason for the modification. Despite this common purpose, each team was found to use the SPR system in a different manner. While e-mail data described the social network on both teams and source control data described the technical network on both teams, these SPR data are especially interesting because they act as a bridge between the social and technical networks. Collocated Developer 2 pointed out that this variance in the data may also be significantly impacted by the testing manager on each team, who traditionally manages the updates of the SPR database. He suggested that further study may validate the findings here, but without such study it is important to acknowledge the powerful role played by one individual on each team. The charts below document the number of individuals modifying the state of SPRs, the number of SPR state changes per week and the average time to resolution for an SPR. The finding resulting from the data described below is that the distributed team has adapted the SPR system as a means for using technology to track work item status.

SPR Actions Per Week

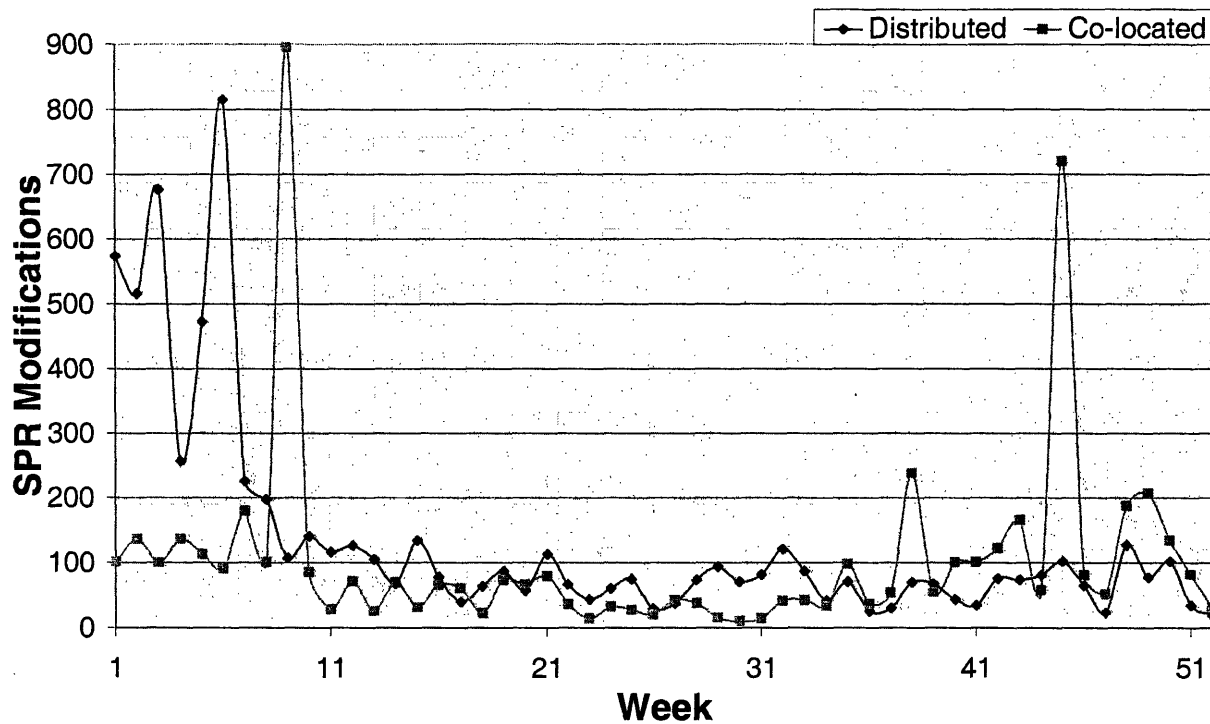


Figure 12. Number of SPR actions per week.

This chart is most interesting because it documents the weeks which had unusually high SPR activity for each team. An SPR action documented in this chart refers to any status update provided to the state of the SPR. For example, a member of the team can designate an SPR as verified, reproduced, fixed, tested, or needing more information. The SPRs also contain a field for comments, and it is common for developers to enter comments on other developers' SPRs, but these were not included in this study because they do not represent an action taken on the SPR. Further study may find merit in examining these comments because they represent another form of contextual collaboration between the team members similar to e-mail.

Without counting all weeks with SPR modifications above 200, the averages were within 10% – the distributed team averaged 76.49 SPR actions per week and the collocated team averaged 70.13. The outliers in the data for the distributed team occurred between weeks 1 and 6.

Both teams shipped a final product release in the first quarter of the year, and thus this high period of activity in the early part of the year represents the SPR “clean-up” activity which occurred for the distributed team as they were updating status on all of the work which had been done for the release. For the collocated team, the major outlier in the data occurred in weeks 9 and 45. These outliers demonstrate that the collocated team required a major focused one-time cleanup immediately preceding the product release, and immediately preceding the feature freeze date. As a participant in this study, I speculate that these one-time cleanup efforts represent an automated process of noting all SPRs which were not fixed before a particular milestone and updating status to indicate that they will not be addressed. The outlier in week 45 essentially represented a mid-stream correction for the collocated team, most likely because the SPR status was not relied upon as heavily to track status during the actual work period before the milestone. The next two charts provide more justification for this speculation.

Average # of Individuals Modifying SPR State (each SPR accounted for in the last week a modification was made)

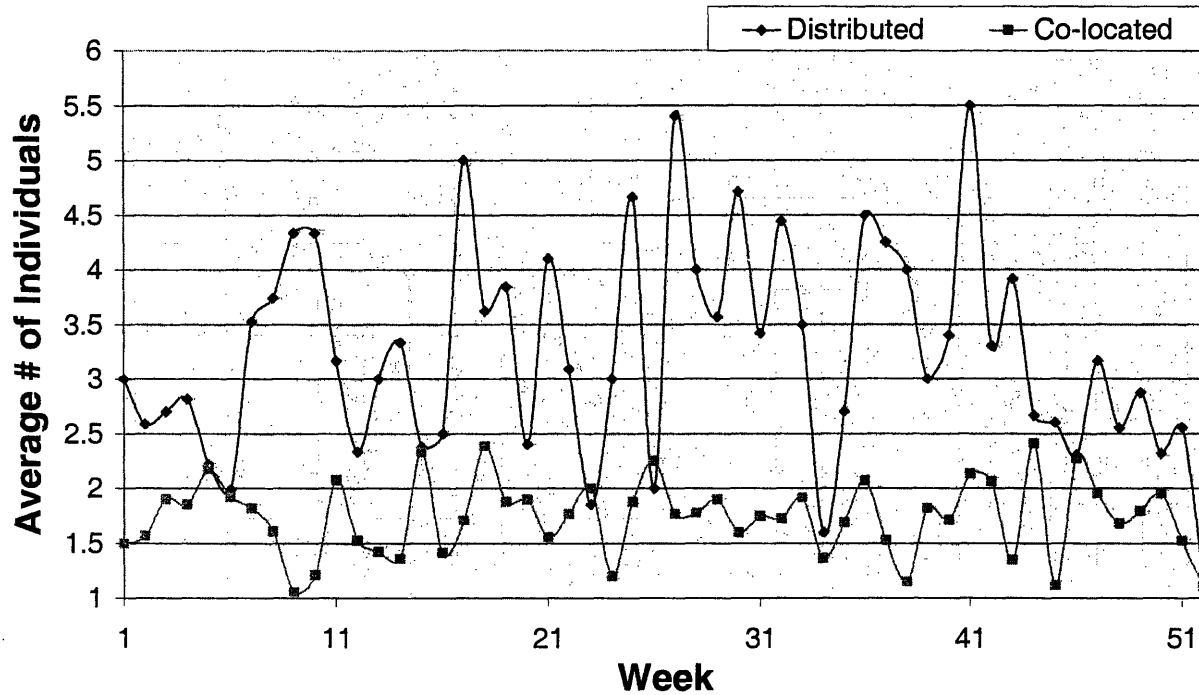


Figure 13. Average number of individuals modifying state of SPRs.

This chart shows that the average number of individuals modifying the SPR state for a given SPR was higher for the distributed team than the collocated team. As with the e-mail thread charts and the source control charts, SPRs are accounted for in the last week for which a modification exists and thus the time series nature of the data is not as relevant as the overall results. The average number of individuals modifying an SPR for the distributed team was 3.25 and for the collocated team was 1.74.

This reinforces the notion that the SPR database was used by the distributed team as more of a collaborative knowledge sharing and status tracking mechanism than for the collocated team. It provides a clear example of the impact of asynchronous work schedules – when team members are not available to consult with immediately, it becomes useful to add updates to SPRs in the context of the particular issue and wait for a reply in the form of another action taken on the SPR.

With the collocated team, since answers are available immediately, it is not useful to take the time to update the formal SPR system when sharing knowledge around a particular SPR.

This demonstration of adapting available technologies in different ways has positive and negative points which are developed further in the Emerging Themes section that follows. The positive aspects are that the team has naturally innovated and found new uses for an existing infrastructure. However with this innovative use comes the caveat for managers that tracking results on a system such as the SPR system will not yield similar reports for teams which use the system differently. Furthermore, it is interesting and important to point out the difference between this data and the data obtained from the source control system. The collocated team had more individuals modifying particular elements of the source control system while the distributed team had more individuals modifying particular elements of the SPR system. This suggests that there are certain thresholds for collaboration and different geographic structures can lead to different levels of social and technical collaboration.

SPRs - Average Time to Resolution

(each SPR accounted for in the last week for which a modification was made)

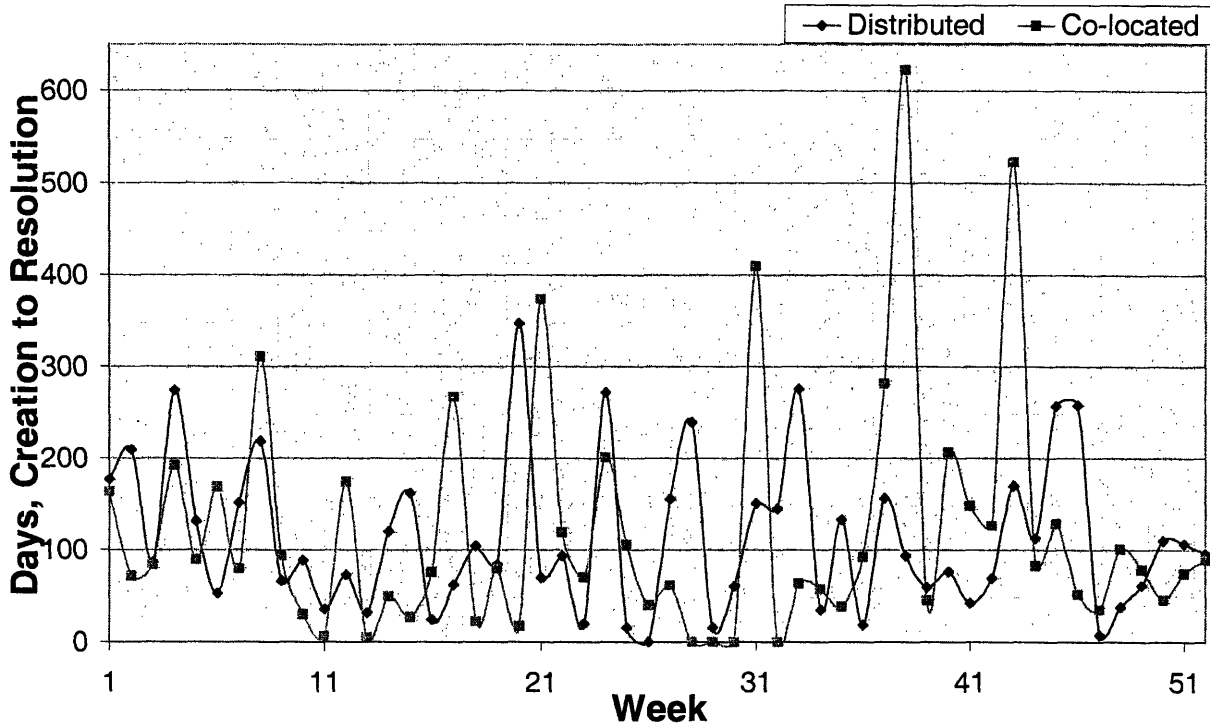


Figure 14. Average time to resolution for SPRs from creation date to resolution date.

This chart is provided as one way of measuring some level of output from each team. It represents a calculation of the total number of days from the point at which an SPR was created to the point when the SPR was marked as Resolved in the database. The average for the distributed team was 113.80 days while the average for the collocated team was 120.72 days. Given the wide range in data, the averages are not as useful to study as is the variance. Similar to the team meeting data, there is a variance in this chart between short and long term issues being resolved, on a week to week basis. This is seen in both teams, and should be an area for future research. It may be the case that teams alternate between focusing on short term SPRs such as customer fixes and minor bugs, and long term SPRs such as features. Future study should build on this chart and determine ways of measuring output from teams so that the observed data can be discussed in the context of positive or negative results.

Informal Communication: Impact on Trust, Creativity and Problem Solving

A number of developers on both teams acknowledged the impact of informal communication on the knowledge sharing process. The developers were asked to describe communications which occur outside of the formal framework of meetings and group e-mail. Overall, the collocated team cited much more informal communication than the distributed team. Collocated Developer 2 stated that "programming is very personal to me – it's like a garden, everything is in disarray if it is not planted with care." He indicated that the informal face to face decision making allows him to see the emotions, gestures and tone of voice of the others, and build trust between his colleagues that their input can be gathered without criticism. An example cited by Collocated Developer 3 of social interactions turning into work discussion was when he was at the coffee machine, saw another developer and was able to casually mentioned a request for help he had sent to this developer earlier. The chance interaction was able to act as a "tap on the shoulder" which would have seemed much harsher if it had come via e-mail. Collocated Developer 1 noted that even having lunch together had a significant impact on team productivity because it boosts morale, and also helps team members realize when others had input on the issues they were trying to resolve. His preference for face to face work styles was described as "a little old school," and likely based on his generation's lack of preference for instant messaging and electronic communication, to the point where he would choose a job based on whether he would be working with collocated teams or not. Solutions for building informal communication on a distributed team such as using instant messaging technologies late at night would not work for him because "nighttime is family time" and altering work hours is not an option. Collocated Developer 3 said that even on a collocated team, the informal communication can be based on geographic distance – during the project, his work area was moved to share an office with

another developer, and he found himself collaborating with this developer to a much larger extent even though it was only a matter of a few yards walk before.

The distributed team did not cite as great a use of informal communication, mostly due to the geographic separation and time zone differences. One key difference was found with Distributed U.S. Developer 1, who was the only developer on the team who had traveled to India to help with the initial work distribution. He stated that the one time face to face meeting allowed him to develop relationships which went beyond the working relationship, and now even when distributed he is able to feel comfortable asking the Indian developers about things such as their family life or social events, which then leads to a greater comfort when requesting help on certain issues. This developer felt that it was possible to build this sort of interaction into "the heart and soul of the group" but that it required much individual effort rather than management direction and particularly the efforts of "one spark plug in every group." The effect of this "spark plug" was seen in the Indian team, where Distributed Indian Developer 1 stated "I always thought Americans were very businesslike and serious – it was a pleasure to connect with someone on a business level." Distributed U.S. Developer 3 described this developer as the "social hub" of the team, because "it's the kind of guy he is." Distributed U.S. Developer 2 pointed out that the Indian members of the team shared a greater deal of informal communication with each other because they were all collocated, and the effects of this were apparent in the feedback they provided during meetings and online discussion forums. It was also interesting to note that the only U.S. developer to refer to Indian developers by name during the interviews for this study was the one developer who had met them face to face.

Individuals on both teams suggested that they were much more creative in informal discussions. Distributed U.S. Developer 1 explained this phenomenon by stating that "because

we're often commiserating, we feel that we're solving the problem as a team" and so "you let your mind roll a little bit, and things come up that might be stupid and get shot down, but it's ok because you're talking among friends." Each developer on the collocated team acknowledged the significant work which is done in the hallways in the few minutes following a formal meeting. Collocated Developer 3 stated that because there is a "strict agenda" in the meetings, the "free flow of ideas which is most valuable in a development group" is seeded in people's minds during the meeting but then is discussed outside the meeting because "you don't feel there are ramifications." According to Collocated Developer 2, "if someone is not in a close working relationship, I feel I have to organize my thoughts before discussing any issues with them" and this is a significant barrier to communication which prevents such informal interaction from occurring on distributed teams. Collocated Developer 3 speculated that one way to encourage this informal communication on a distributed team would be to form "atomic units" where smaller groups of developers would be forced to work on overlapping technical areas, so the frequent nature of communication would cause social relationships and trust to build.

The issue of culture was not a focus of this study, but it did surface in some of the interviews as an important factor to incorporate into understanding the nature of informal communication. U.S. and Indian developers take different levels of familiarity to be able to communicate informally, stated Distributed U.S. Developer 2. Additionally, Distributed Indian Developer 3 stated, "failures in communication often occur due to implicit assumptions made by individuals, sometimes cultural in origin - this could be a cause for confusion since both sides are left wondering why the other 'did do' or 'did not do' what to them is obvious." The role of culture in the dynamics of a geographically dispersed team is a fertile area for further study, but

should be acknowledged in any study such as this which looks at factors influencing geographically dispersed teams.

Emerging Themes

A number of cross-cutting themes emerged from the data analysis. In lifting these themes from the analysis, it becomes clear that the lessons from these data need not only be applied to cases where geographic distribution is a factor.

Same Technologies and Processes can be Applied Differently

The two teams clearly had very different ways of applying the same, or very similar, technologies and processes. Electronic mail was used on the distributed team much more for discussions, while on the co-located team the use of mail was restricted to announcements or very short discussions. Similarly, the higher level of modifications to the software problem reports in the distributed team suggested that they use this technology as a means of transferring information between team members and maintaining a record of status, while the co-located team could rely on synchronous communication for information sharing and status reporting. In the case of meeting minutes, while the agenda categories were generally the same between meetings, the number of tactical items and number of task assignments were much higher in the distributed team. The distributed team used the meeting process as a means for addressing needs that the co-located team satisfied outside of the context of the formal weekly meeting.

These differences suggest that technology and processes which support knowledge sharing can be built and implemented to explicitly serve different purposes. Barley provides a framework for assessing technology's role in a knowledge-based work environment and suggests that the context in which the work is performed can significantly impact the way the technology is used⁷⁷. Teams would be well served to assess their own uses of technologies and determine if explicit changes need to be made to the particular technologies which are being used. For example, extensive use of electronic mail for discussions may warrant a greater use of discussion

databases which are designed for long threaded discussions, while use of electronic mail for announcements may warrant a greater use of instant announcements over instant messaging systems or a dynamic team web site where announcements may be posted.

The differences also suggest that managers should be cognizant of differing uses of technology and process when assessing their own team's progress. Managers need to adjust their expectations on the level and style of communication based on the geographic structure of the team. Collecting data similar to what was collected in this study to provide a "dashboard" for managers to continually assess the level of collaboration and progress on their team could be very useful. However, misaligned expectations could lead to gross misinterpretations of this data. For example, if a team accomplishes most of its tasks in an ad hoc and informal manner, periodic monitoring of a system such as the SPR system may indicate that the team is making much less progress than it actually is. Thus, it is important to understand the normal use of technology and processes and then adapt expectations appropriately. The development manager of both teams acknowledged this need for adapting management practices to each team. He stated that when he first took over the collocated team, he had been used to the processes of the distributed team and "that didn't work well for the team." He has since altered his use of e-mail, in person visits and meetings to adjust to the differing needs of each team, and found much more success in the resulting knowledge flow on each team.

Social Relationships and Technical Productivity are Intertwined

The degree of social interaction between developers on a team was shown to have an impact on the technical productivity of the team, which then led to tighter social relationships. Developers on both teams cited the comfort level between team members as important in facilitating creative discussions, where developers do not have to worry about feeling

embarrassed by a poor idea. The source code data showed that the co-located team had many more examples of code elements which were modified by multiple team members, and interviews confirmed that this was likely because of the greater degree of social interaction on this team, rather than any technical reason that one product merited more intertwined technical interaction than the other. The interviews also suggested many cases where a casual or informal interaction led to a technical decision being made, as described in the earlier section. While such social relationships are much easier to form when colocated, the experiences of the one U.S. developer on the distributed team who traveled to India suggest that social relationships can be built across geographic borders and these relationships can indeed be leveraged to satisfy technical goals.

Teams React Differently to Different Stages of the Project

Managers need to be aware of differing patterns of activity which may surround the milestones of a project, and set expectations appropriately given the geographic distribution of the team they are managing. The data showed that each team exhibited a different pattern of activity at the different stages of the project schedule. In the first three months of the study, as the teams approached a final product release, the distributed team spent 100% of their meetings discussing tactical issues, and had significant SPR activity. In the next seven months of the study the teams approached "feature freeze", which represents the major milestone for software developers during a release when code for all features in the new release is submitted into the product. As this milestone was passed, the colocated team had a lot of coordination to do with regard to the source code, and also saw an increase in e-mail activity which was not there during the period before the milestone. None of these behaviors were cited in the interviews as

particularly positive or negative, they were instead seen by each team member as the regular reaction to the different stages of the project.

Structure Does Not Define Destiny

The geographic structure of the teams in this study led to different forms of value being achieved from their knowledge sharing processes, however it is not necessary that this destiny for each team be defined by structure. The chart below illustrates a highlighted example of this theme. In the chart, the structure of the distributed team led them to have a higher degree of documented decisions – this is shown in the data through increasing uses of emails, tactical meeting items, and SPRs. The interviews with members of this team confirmed that one of the pieces of value obtained by this process was that the history of decision making was better retained. Interviews with the co-located team indicate that it would not be feasible to enforce the same level of documentation on the collocated team, however this need not be seen as a barrier to achieving the cited value of history retention. Instead, alternative processes such as scheduled time for documentation or the implementation of automated documentation tools could be used to achieve the same value for the collocated team. Similarly, the collocated team cited the informal communication as a process which led to higher degrees of creative and new solutions being found. Even though these informal meetings generally occurred face to face, the value achieved is something the distributed team can still obtain. Two suggestions provided in the interviews with the distributed team on this specific topic were a one-time face to face meeting which would introduce team members and bring a social component to the relationships, and the use of explicitly informal phone calls where no topic is predetermined so there is the potential to discuss any open ended topic.

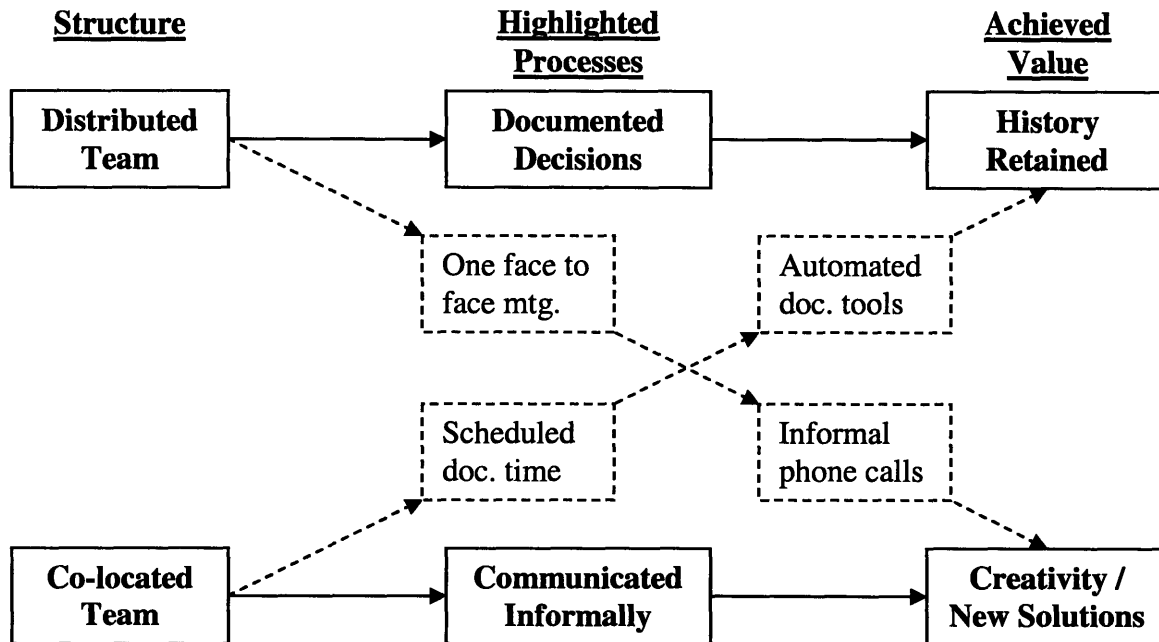


Figure 15. Example scenarios where the structure of the team leads to a process which achieves a certain value that can then be achieved in a different manner by the team whose structure does not allow for the highlighted process. Thus, the tight link between a structure, process and value does not need to exist.

The examples discussed above highlight the more general theme that structure does not indeed define destiny. Organizations with teams having different structures should make explicit attempts to understand the value being obtained by different structural formations and then determine ways to adapt the processes from one structure to the other to achieve the same value.

Individuals can Heavily Influence Coordinated Development

Individual behaviors on a distributed team can have a significant impact on the success of coordinated development efforts. US Developer 1 on the distributed team made a significant number of contacts on the India team which led his collaboration with them to be much more frequent and useful. The project manager who kept minutes of meetings from both teams was very meticulous in the description of meeting items which led to better history retention on both teams and also more accurate data collection for this study. Another example is Developer 4 on

the collocated team, who cited that he strongly prefers collocated teams and has adapted his personal work style to fit the collocated model. This is an example of an individual who has recognized the individual effort needed to work in a distributed or collocated manner, and has identified which mode is appropriate for them. In all of these cases, the behavior of an individual was not predetermined by the geographic structure of the team, and had a significant impact on the success of their team.

Geographic Distribution can be an Asset

The parallel nature of the data – with positive findings on collaborative successes from each team – suggests that the common theme in the literature of geographic distribution being a barrier to overcome is not sufficient. Instead, geographic distribution should be seen as a potential asset which can be leveraged, along with time zone differences. Numerous benefits from leveraging the geographic structure were cited in the interviews with the distributed team. Example of this include an increase in documentation and history retention; the ability to share short term tasks which required immediate attention so that work could be performed around the clock; and a more structured definition of work tasks and distribution of work items. The interviews cited a common understanding that many of these behaviors were a natural reaction to the geographic structure, however in retrospect the value of these behaviors was realized. The core theme is that when managing a distributed team, the distribution should be looked upon as a characteristic to leverage rather than a barrier to overcome.

Technology and Policy Impact Assessment

While this is just one case study of one pair of teams, the variance in the data between teams allows speculation on the technology and policy impacts of geographic distribution of knowledge-based teams. In the context of this research, we define technology and policy solutions as those which combine changes in the technology (both technology being used to support the project and technology being produced in the product) and changes in the policy (both organizational policy and public policy). This section discusses proposals which address various levels – individual, team, organizational, institutional and national. The proposals described in this section can be validated with more in depth research. The core finding here is that an innovative use of archival data as done in this study can provide input into the technology and policy decision making process.

Individual

Individuals who work in a knowledge based industry have the power to make choices about the global working environment. Geographic distribution of some groups creates another dimension on which individuals can choose which projects they work on and which technologies they want to develop. Additionally, being successful in either a distributed or collocated environment is a capability which individuals can invest time and effort into developing. The data in this study suggests that methods of success can differ based on geographic structure even when all other aspects of the working environment and technology are controlled. The data suggests that there is no right choice to pursue, and neither structure is more challenging than the other.

Individuals also have the potential to use software tools similar to those used in this study to educate themselves on the work of their own teams. For example, it would be useful for

individuals to have an understanding of how knowledge sharing technologies are being used as different parts of the project lifecycle arrive so they can be better prepared to achieve their own knowledge gathering activities. Access to the data analyzed in this study is a significant issue because if it is not stored in a convenient format or made accessible via an easy to use tool, it will not be useful as a tool for individuals.

Finally, on the individual level, the data suggests that workers may find benefits in adjusting their assumptions about work hours and styles to fit new geographic structures. The value of synchronous communication was identified in the interviews – both for building social relationships as well as sharing knowledge more rapidly. Thus, it may be beneficial at times for individuals in a distributed environment to alter their work hours to spend a few minutes in their off-hours to either use the telephone or instant messaging to communicate with colleagues working in a different time zone. A change in work styles may require more effort to be placed in explicit informal communication on a distributed team, or in more formal documentation of informal decisions on a collocated team. Any of these changes requires an understanding of the assumptions which exist with each working environment and an appreciation for the value which can be achieved by moving beyond these assumptions.

Team

On the team level, an understanding of the socio-technical forces which impact knowledge sharing on a software team can impact the success of both the project and the product. The system level diagram produced from the expert seminar conducted as part of this study lists many of the factors which can impact a team when operating in a geographically distributed environment. These factors should influence decisions on which processes are followed for

knowledge sharing and how the technical elements of the product are split between team members.

In making decisions on team structure and technical architecture, software tools which continually collect and display information similar to the data collected in this study will be useful to inform managers. Building and implementing such tools would provide access to data which resides in distributed and heterogeneous sources and is not currently used by managers to guide decisions. One example of the utility of such a "dash board" is in assessing the technical dependencies between members of a team by looking at the SPR and source control data. If a manager had an idea of a normal level of collaboration for the team on each of these aspects, it would be possible to identify situations where one member of the team is interacting with a broad spectrum of issues and adjust by assigning different team members to the different parts of the product. Another example is that of a manager who monitors email thread activity during the different stages of a project lifecycle, and can observe the level of activity as a milestone approaches. If the email thread activity is greater than normal the manager can put extra attention into determining whether the team requires additional meetings or technical load balancing to return to a normal state of knowledge sharing.

Finally, both teams in this study were at a mature stage in the product lifecycle and had very specific technical tasks. There is a need to extend this analysis to teams of different sizes, stage in the lifecycle, and technical domains. Understanding teams along these different dimensions will help validate the utility of such data collection in guiding a team and will also provide additional perspective on which all teams can base their own decisions. The stages of team development identified by Tuckman can be used to categorize the data and analyze accordingly⁷⁸.

Organization

At the organizational level, the data collected in this study demonstrates the potential for organizations to assess the tacit knowledge capital which is not readily quantifiable. Organizations can currently assess knowledge capital by counting the number of patents filed or tabulating features on existing products, but with the data capture tools used in this study, organizations can assess knowledge capital at a much more granular level. It now becomes possible to assess the dependency on one development site or another, and on one developer or another. This not only allows an aggregate tabulation of knowledge capital, but also allows the flow of knowledge capital to be tracked. This is especially important in a domain such as software engineering where knowledge flows so dynamically between geographic locations. If organizations have to make decisions on how to move work between locations, technology which provides a snapshot of knowledge flow provides vital input.

The impact of this study on the organization can further be understood if viewed from three lenses – structural, political and cultural. The structural perspective has already been discussed in great detail on the team level, but on the organizational level this study suggests that an organization's structure does not have to be consistent across teams or technical domains. Furthermore, an organization's processes and standards must be flexible enough to be adapted to each of the team structures it supports. The political perspective on this issue is that exposition of data as granular and revealing as the data provided in this study has the potential to expose where the true decisions are made within a group. For example, whereas on the surface it may appear that decisions are made in meetings, it may become apparent through the data analysis that individuals on the team reach consensus through informal email discussion and then use the meetings as a formal approval mechanism. From the cultural perspective, an organization's culture is greatly influenced by its geographic structure and the ability to understand the links

between these two facets is tremendously powerful. On the surface, this study was not an examination of contrasting regional cultures, but the impact of regional cultures on organizational cultures does exhibit itself through the data. For example, it is possible to analyze the proclivity towards strict processes or the tendency to make decisions in ad hoc brainstorming sessions. Thus, the interrelated structural, political and cultural dimensions of organizations are all profoundly impacted by the introduction of knowledge sharing data.

Institutions

A number of institutions are impacted by the introduction of granular knowledge sharing data analysis in global software teams. The laws and regulations which govern labor and trade are not yet built to handle redefinitions in work requirements and intellectual property sharing which occur instantly and cross geographic borders. Unions and professional associations which represent one region of technical knowledge workers can act as both enabling and threatened institutions. They can enable the success of global software teams by training their members to build the capabilities necessary to operate in a geographically distributed environment, and also by expanding their regional scope so that their incentives are aligned with those of the multinational firms which employ them. However they can also act as a threatened institution by deciding that the work models, regional job opportunities and technical expertise requirements are all being negatively impacted by the distribution of work. In all of these cases, the notion developed in this study that knowledge sharing can be analyzed and leveraged is important in assuring that institutions impacted are able to adapt appropriately.

Nation

On the national level, this study develops a number of methods which may be used as inputs into major public policy decisions around the globalization of knowledge based work such

as software development. It is not the goal of this study to take part in the traditional debate to understand and debate the pros and cons of offshore outsourcing. This study suggests that a sustainable model is possible, where knowledge is dynamic and our nation can innovate and remain a player in the global software industry on a permanent basis. The National Innovation Initiative is one example of the direction of national public policy to improve the value added by U. S. engineers by investing in innovation⁷⁹. Policies such as this require a sustainable model for the innovation to be incorporated into the global manufacturing chain for software if certain lower-skill, lower-paying jobs must be done offshore. The data presented in this study suggests that dynamic models are possible which can adapt the knowledge sharing to the locations in which the different types of work are being done.

The individual traits for success in the global software engineering domain discussed in this study can be used to develop national policies on training and preparing U.S. workers for globally distributed work. The implementation of this policy may involve encouraging organizations to train their employees in methods which expose the knowledge they possess, or in reaching the knowledge workers themselves. The key point is that national policy needs to incorporate some form of enablement for U.S. workers to be able to make the most of the knowledge and abilities they possess.

Additional impacts on national policy from this study revolve around the need to fully understand the level of intellectual property being shared around the world. Intellectual property trade regulations and export controls have the potential to fundamentally affect the nature of geographically distributed work⁸⁰. With every nightly communication among the members of the distributed team in this report, intellectual property was being exported from one country to the other. If public policy existed to valuate and regulate exports at this level of granularity, the

ability of nations to exploit wage differences would be significantly limited. Additionally, the process and documentation costs of such granular monitoring of IP export would be a significant factor to be incorporated. The significant jump in this issue provided by this study is a means of exposing inner-team knowledge so that IP valuation does not have to be conducted using only tangible items such as patents, licenses and product releases. The other reason that understanding intellectual property flow is important on the national level is that it gives U.S. policy makers an understanding of the impact of trade regulations which have an impact on relations with foreign countries that are motivated by non-financial reasons. For example, if a political disagreement with a foreign country causes the U.S. to discourage trade with that country, it would be beneficial to have an input to this decision be the level of IP currently invested within that country.

This research does not need to solely apply to the United States. The themes discussed here illustrate how any country can succeed or fail in the global knowledge economy.

Future Research Directions

The nature of this study, as an innovative means of using archival data to assess knowledge sharing in one type of global software team, lends itself very well to future research in many dimensions.

Larger Number of Teams

The first obvious dimension of further study would be to study a larger number of teams. Maintaining the same controls on team size, scope and product, it would be very useful to collect data on a large number of teams to validate the findings' relevance to the geographical structure of the team.

Results of Weekly Feedback to a Team

If teams received weekly feedback related to the data which was collected that week, the behavior of the team may change to adapt their work to what they learn. The interviews conducted in this study were done before and after the data collection, and in discussing the data results with team members, it was clear that some insights were not obvious to team members and would cause definite changes in interaction. Providing feedback to one team while not providing it to another would also provide an interesting experimental result which may validate the need for better technology to educate teams on their own knowledge sharing practices.

Collection of Other Data

The diagram early in this report which described the inputs, processes and outputs of a software development team included a number of other potential sources of data which would provide a more complete story of the knowledge sharing on software development teams. Examples of this include instant messaging usage, phone usage, development schedules and logging of informal interactions.

Varying Types of Software Teams

It is relevant to expand this study into other types of software teams. Teams which are a different size, in a different technical domain, responsible for a product in a different stage of the lifecycle or in different companies are all worth studying. There are also different functions of software development teams beyond producing product releases. Software development teams can act in a consulting role, producing short-term solutions for one customer, or in a support role, solely producing fixes on an existing product in market. Each of these different types of teams has a different type of knowledge to share and different methods of sharing the knowledge. Applying the data collection techniques developed in this study to those teams would provide insight into how geographic distribution affects varying forms of knowledge sharing.

Other Knowledge-Based Industries

Many other industries beyond software development are engaging in distributed team work, and archival data in these teams can yield similarly powerful results regarding the impact of geographic distribution. Directly comparable industries such as mechanical engineering may provide insight that can be incorporated back into the findings related to the software industry. Additionally, industries where work occurs in a time-shifted basis may also benefit from similar study. One developer on the collocated team compared software development on a distributed team to the practice of nursing – when a nurse completes a shift, they have to transfer the knowledge obtained in that shift to the next shift of nurses, and this is often done in a hybrid form of structured and unstructured communication mechanisms. Industries such as nursing are not directly related to the product design and engineering domain of software engineering, but still provide fertile ground for future research.

Technology Prototypes

The next step in applying these findings is to build technology prototypes which provide access to the data collected in this study. Such prototypes would be useful in acting as a "dash board" for team managers to monitor social and technical collaboration on their teams, as well as providing useful predictors on the team's progress towards goals or milestones. The technology prototypes could also be used to educate individual team members on their team's collaborative processes, and also on the processes of other teams. Finally, technology could be built to collect even more data than was possible in this study. Much of the data collected in this study was in inconsistent formats and required a significant degree of manual cleaning before being incorporated into the analysis. If technology was built which was able to parse through much more diverse sets of data such as web discussion forums, document revisions and comment lines within source code, an even richer picture of the knowledge sharing on global software teams can be obtained.

Conclusion

This study exposes many of the impacts of geographic distribution on the socio-technical system of software development, and demonstrates the power of archival data in providing insight which can be used to guide decision making on knowledge based teams. The data collected in this study revealed that each team found an appropriate mode of operating, on both the social and technical levels, and in many cases used similar technologies and processes in very different ways. When combined with the system level context provided by a series of expert stakeholder meetings, a number of technology and policy proposals emerged which all shared the common theme of not allowing geographic distribution to bring along certain assumptions about the way knowledge based individuals and organizations should operate. Data collected in this study provided a solid foundation for future research into many areas of global software development, as well as providing a suggestive framework for technology prototypes which can be used to provide teams with a greater awareness of their own knowledge sharing practices. The potential impact of such tools is astounding – not only may they help individuals, teams and organizations better utilize their own knowledge capital, but they may also serve to help nations understand the intellectual property being traded across global boundaries and have significant implications on trade and export regulation policy and international taxation.

As a participant-observer in this research study, many of the benefits and caveats originally identified when choosing this method came to bear in the data collection. Much of the data collection and data cleaning was a manual process – inspecting meeting minutes, extracting data from proprietary SPR and source control databases and deciding which questions to ask during interviews. These processes were much more efficiently handled because I had experience as a member of the team and could understand the terms and idiosyncrasies and also

had a level of trust with colleagues on both teams. Many times in the interviews, I found that even a gentle nod from myself conveyed to the interview subject that I understood and shared the situation they were describing, and I had a level of trust which an outsider may not have enjoyed. Further study from others in this field will reveal the limitations that my role as a participant had on the study. There may be pieces of the data which may have been more salient to collect than the choices I made, and the critique of both teams may be harsher from an outsider, but as long as these caveats are well understood, this study holds merit in making innovative use of the archival data on both teams.

Many other knowledge-based industries serve to benefit from similar study, as knowledge-based work becomes more global. The biotechnology industry is growing; however the lab based structure of the research in this industry suggests that even more innovations will be required over and above what has been done in the software industry. There is even an impact to be had in industries which are simply time shifted rather than geographically distributed. Collocated Developer 2 suggested a comparison between geographically distributed development and the shift style work of nurses. When a nurse leaves their shift, they have to communicate all of their knowledge related to a set of patients to the next shift of nurses. They have to make decisions on which pieces of information to include in the formal mechanisms such as patient history charts and which to discuss in person.

In closing, I believe this study demonstrates that careful nuanced management of global teams will result in powerful success regardless of the structure which is employed. A good manager will be able to determine which tasks and which products are best suited for a given structure, and will be able to provide this input to the organization making decisions on global work location. Intelligent policy makers are best served to understand this notion as well, so

they can build policies which support a global framework for knowledge based work while preserving opportunities for workers in their constituent geography.

Appendices

Research Prospectus Submitted to IBM

Project Overview:

“Communication in Global Software Development Teams”

A Case Study at the IBM Software Group – Lotus division Conducted as part of the MIT Technology and Policy Program (TPP)

Project Summary:

This project will explore the communication mechanisms utilized in two software teams at IBM. One of the two teams is divided between the United States and India; the other team has all but one of its members working on the same hallway. Other than this, the teams are very similar, as they have the same development manager, same project manager, similar product scope, lifecycle stage and schedule, and develop very similar technology. The goal of this project is to understand what factors impact the knowledge sharing in these two teams and how each of the team structures affects the choice in communication style and substance. Each team's structure has potential benefits – for example, the geographically dispersed team can work on tasks “around the clock” as they are shared between developers in different time zones, while the co-located team can work face-to-face around a whiteboard.

Key Research Questions:

- How does informal communication impact the technical progress of a software team?
- How does asynchronous (24-hr) development compare with co-located development with respect to performance to schedule, time to resolution and software quality?
- What factors improve the organizational learning of a small software team?
- How do cultural differences impact a software team's process and communication?
- How important are the various forms of communication in each team environment?

Research Design:

The research will involve an analysis of existing team data and interviews of team members on both teams to compare social, technical and organizational relationships. Tools will be created to aggregate software problem report (SPR) history and source code change history to document the nature and level of interaction between team members, as well as form a network map of team members. Relevant e-mails sent between team members will be analyzed for thread depth to gain additional insight. Meeting minutes of the US-India team and the US-only team will be analyzed for key differences in communication levels. Finally, short weekly interviews of each team member will document informal and instant message communication during the week, as well as report feedback to the team member on the ongoing findings. Throughout the process, there will be respect for confidentiality and a commitment to constructive shared learning.

Expected Yield:

A TPP masters thesis on software team communications, as well as potential scholarly articles or learning materials on this topic. Feedback briefings will discuss findings with the teams involved.

Organizational Impact and Assurances:

The interviews will minimize the time required of participants to only a few minutes each week, over the two month period. A guiding focus will be to foster constructive learning within the teams. The interviews will be conducted on a voluntary basis. The identities of individuals and teams involved will not be disclosed in the thesis or any associated presentations or articles. No proprietary information will be included in the thesis or any associated presentations or articles.

Project Team:

Satwik Seshasai will conduct the research, as a graduate student at MIT, as well as a developer on the Team Workplace team. Dr. Joel Cutcher-Gershenfeld of MIT will serve as the advisor.

Interview Protocol Submitted to IBM

Interview Protocol:

“Communication in Global Software Development Teams”

**10-15 Minute Weekly Discussion with Software Developers
Conducted as part an MIT TPP Research Project at IBM**

Interview Introduction:

Thank you for taking part in this interview. Our goal is to gain an appreciation for the intangible factors related to communication on a software development team. All of the questions are voluntary and names will not be included in the final report. We will try to limit the time of the interview since it is a weekly discussion. Shall we begin?

Questions to be asked in initial session (30 minutes):

- What is the frequency and nature of interactions between team members?
 - Both co-located and geographically dispersed
 - Methods of interaction: e-mail, IM, phone
 - One on one, or group?
 - Formal or informal?
 - Role-based or not?
- What is the past history of this team?
 - Recently introduced geographical dispersion?
 - How did interaction change from before to after?
- What is the organizational structure of this team?
 - Which tasks are geography based and which are project/team based?
 - Are there differences between reporting structure vs. operational structure?
- Is most work activity-based or team-based?
 - Are ad-hoc “teams” formed around particular activities?

Questions to be asked in weekly sessions (10-15 minutes):

- Social network
 - Who did you talk to this week, formally (i.e., “because you had to”)?
 - Who did you talk to this week, informally (i.e., without a specific job task in mind)?
- Performance to schedule, time to resolution, software quality, feature specifications
 - Are there any particularly significant impacts to the above topics that you can attribute to your team’s geographic/temporal distribution (either co-located or dispersed)?
- Cultural differences
 - Are there examples of cultural similarities or differences between you and your co-workers that you felt affected your communication ability, either positively or negatively?
- Synchronous communication
 - How often did you use instant messaging to communicate with team members (in relation to email/phone, during non-regular hours, and for in-depth vs. quick questions)
 - How frequently did you interact face-to-face with your team members? Were there particular interactions which would have been much less effective over email/phone/IM?
- Infrastructure
 - Has the location of the build system changed the manner in which you collaborated?
 - Has the location of the source control system affected your collaboration?

Interview Conclusion:

Thank you very much for participating in this study. If there are any additional issues which you remember in the next few days, please feel free to contact me.

References

- ¹ Herbsleb, J. D. et. al. "An Empirical Study of Global Software Development: Distance and Speed." Proceedings of the 23d International Conference on Software Engineering, Toronto, Canada, 2001.
- ² Haag, Z., R. Foley, and J. Newman. "A Deontic Formalism for Co-ordinating Software Development in Virtual Software Corporation." Proceedings of WET ICE 98, IEEE Press, 1998.
- ³ Seaman, C. B. "Qualitative Methods in Empirical Studies of Software Engineering." IEEE Transactions on Software Engineering, Vol. 25, No. 4, Jul/Aug 1999. pp. 557-572.
- ⁴ Poltrock, Steven E. and Jonathan Grudin. "Organizational Obstacles to Interface and Development: Two Participant – Observer Studies." *ACM Transactions on Computer-Human Interaction*. Vol. 1, No. 1, Mar. 1994, pp 52-80.
- ⁵ Meyer, Mary A. "How to Apply the Anthropological Technique of Participant Observation to Knowledge Acquisition for Expert Systems." IEEE Transactions on Systems, Man, and Cybernetics, Vol. 22, No. 5. Sept/Oct. 1992.
- ⁶ Burns, C. M., and K. J. Vicente. "A participant-observer study of ergonomics in engineering design: how constraints drive design process." *Applied Ergonomics*, Vol.31, 2000. pp. 73-82.
- ⁷ Myers, M. D. "Investigating Information Systems with Ethnographic Research." *Communications of the Association for Information Systems*. Vol. 2, Art. 23. Dec. 1999.
- ⁸ Elliott, M. S. and W. Scacchi. "Free Software: A Case Study of Software Development in a Virtual Organizational Culture." *ISR Technical Report #UCI-ISR-03-6*, University of California – Irvine, Institute for Software Research. Aug. 2003.
- ⁹ Taylor, P., Bain, P. "“An Assembly Line in the Head’: Work and Employee Relations in the Call Centre.” *Industrial Relations Journal*, Vol. 30, No. 2, 1999.
- ¹⁰ Drucker, P. "They’re not employees, they’re people." *Harvard Business Review*. Feb 2002, pp 70-77.
- ¹¹ Aron, R., Singh, J. "IT Enabled Strategic Outsourcing: Knowledge Intensive Firms, Information Work and the Extended Organizational Form." The Wharton School, University of Pennsylvania, 2004.
- ¹² Elenkov, D. "Can american management practices work in Russia?" *California Management Review*, Summer 1998, pp 133-157.
- ¹³ Johnson, M. "Learning from toys." *California Management Review*. Spring 2001, pp 106-125.
- ¹⁴ Barhtelemy, J. "The hidden costs of it outsourcing." *Sloan Management Review*. Spring 2001, 60-69
- ¹⁵ Lei, D., Slocum, J. "Global strategy, competence building and strategic alliances." *California Management Review*, Fall 1992, pp 81-97.
- ¹⁶ Tallman, S., Fladmoe-Lindquist, K. "Internationalization globalization and capability based strategy." *California Management Review*. Fall 2002, pp 116-136.
- ¹⁷ Powell, W. "Learning from collaboration: Knowledge and Networks in the Biotechnology and Pharmaceutical Industries." *California Management Review*. Spring 1998, pp 228-240.
- ¹⁸ Davenport, T., Klahr, P. "Managing customer support knowledge." Spring 1998, pp 195-208.
- ¹⁹ Quinn, J. "Strategic outsourcing - leveraging knowledge capabilities." *Sloan Management Review*. Summer 1999.
- ²⁰ Agrawal, V., Farrell, D., Remes, J. "Offshoring and Beyond", *McKinsey Quarterly*, No. 4, 2003.

-
- ²¹ Kaka, N. "A choice of models." *McKinsey Quarterly*. No. 4, 2003.
- ²² Sanuders, C., Gebelt, M., Hu, Q. "Achieving success in IT outsourcing." *California Management Review*. Winter 1997, pp 63-79.
- ²³ Barney, J. "How a firms capabilities affect boundary decisions." *Sloan Management Review*. Spring 1999, pp 137-145.
- ²⁴ Carr, N. "In-praise-of-walls." *Sloan Management Review*. Spring 2004, pp 10-13.
- ²⁵ Christensen, C. "The past and future of competitive advantage." *Sloan Management Review*. Winter 2001, pp 105-109.
- ²⁶ Light, D. "Cross-cultural lessons in leadership." *Sloan Management Review*. Fall 2003, pp 5-6.
- ²⁷ DiRomualdo, A., Gurbaxani, V. "Strategic intent for IT outsourcing." *Sloan Management Review*. Summer 1998, pp 67-80.
- ²⁸ Young, J. "Global competition - the new reality." *California Management Review*. Spring 1985, pp 11-25.
- ²⁹ Mizoras, A. "In House versus Outsourced." *IDC Opinion*, IDC, 2004.
- ³⁰ Champy, J. "Is technology delivering on its Productivity Promise?" *Financial Executive*, October 2003, pp 34-39.
- ³¹ McFarlan, F., Nolan, R. "How to manage an IT outsourcing alliance." *Sloan Management Review*. Winter 1995, pp 9-23.
- ³² Lacity, M. Willcocks, D., Feeny, D. "IT outsourcing: maximize flexibility and control." *Harvard Business Review*. May-Jun 1995, pp 84-93.
- ³³ Lacity, M., Hirschheim, R. "The IS outsourcing bandwagon." *Sloan Management Review*. Fall 1993, pp 73-86.
- ³⁴ Lacity, M. Willcocks, D., Feeny, D. "The value of selective IT sourcing." *Sloan Management Review*. Spring 1996, pp 13-25.
- ³⁵ Quinn, J., Hilmer, S. "Strategic outsourcing." *Sloan Management Review*. Summer 1994, pp 43-55.
- ³⁶ Fuchs, P., Mifflin, K., Miller, D., Whitney, J. "Strategic integration: Competing in the Age of Capabilities." *California Management Review*. Spring 2000, pp 118-147.
- ³⁷ Light, D. "Cross-cultural lessons in leadership." *Sloan Management Review*. Fall 2003, pp 5-6.
- ³⁸ Arnold, D., Quelch, J. "New strategies in emerging markets." *Sloan Management Review*. Fall 1998, pp 7-20.
- ³⁹ offshoring drive.pdf
- ⁴⁰ Millman, G. "Going Beyond Commodity Outsourcing." *Financial Executive*. Sep 2003, pp 55-57.
- ⁴¹ Cheifetz, I. "Think Like Buffet About How to Value Outsourcing." *Financial Executive*. Sep 2003, pp 58.
- ⁴² Chesbrough, H., Teece, D. "Organizing for innovation." *Harvard Business Review*. *Best of Harvard Business Review 1996*, pp 127-135.
- ⁴³ Quinn, J. "Outsourcing innovation the new engine of growth." *Sloan Management Review*. Summer 2000, pp 14-29.
- ⁴⁴ Iansiti, M. Levien, R. "Strategy as ecology." *Harvard Business Review*. March 2004, pp. 68-79.
- ⁴⁵ Begley, T., Boyd, D. "The need for a corporate global mind-set." *Sloan Management Review*. Winter 2003, pp 25-33.

-
- ⁴⁶ Kumra, G., Sinha, J. "The next hurdle for Indian IT" *McKinsey Quarterly*, 2003 Special Edition, No. 4.
- ⁴⁷ Kern, T., Willcocks, L., van Heck, E. "The winner's case in IT outsourcing." Winter 2002, pp 47-70.
- ⁴⁸ Useem, M., Harder, J. "Leading laterally in Company Outsourcing." *Sloan Management Review*. Winter 2000, pp 25-36.
- ⁴⁹ Lovelock, C., Yip, G. "Developing global strategies for service businesses." *California Management Review*. Winter 1996, pp 64-86.
- ⁵⁰ MacCormack, A., Newman, L., Rosenfeld, D. "The New Dynamics of Global Manufacturing Site Location." *Sloan Management Review*. Summer 1994, pp 69-80.
- ⁵¹ Bartmess, A., Cerny, K. "Building competitive advantage through a global network of competencies." *California Management Review*. Winter 1993, pp 78-103.
- ⁵² Venkatraman, N. "Beyond outsourcing: Managing IT Resources as a Value Chain." *Sloan Management Review*. Spring 1997, pp 51-64.
- ⁵³ Seitz, M., Peattie, K. "Meeting the closed-loop challenge." *California Management Review*. Winter 2004, pp 74-89.
- ⁵⁴ Blaxill, M., Hout, T. "The fallacy of the overhead quick fix." *Harvard Business Review*. Jul-Aug 1991, pp 93-101.
- ⁵⁵ Pisano, G., Wheelwright, S. "The new logic of high tech R&D." *Harvard Business Review*. Sep-Oct 1995, pp 93-105.
- ⁵⁶ Porter, M. "Changing patterns of International Competition." *California Management Review*. Winter 1986, pp 9-40.
- ⁵⁷ Earl, M. "The risks of outsourcing IT." *Sloan Management Review*. Spring 1996, pp 26-32.
- ⁵⁸ Granstand, O., Patel, P., Pavitt, K. "Multi-technology corporations: Why They Have Distributed Rather than Distinctive Core Competencies" *California Management Review*. Summer 1997, pp 8-25.
- ⁵⁹ Magretta, J. "The Power of Virtual Integration: An Interview with Dell Computer's Michael Dell." *Harvard Business Review*. Mar-Apr 1998, pp 73-84.
- ⁶⁰ Borchers, Greg. "The Software Engineering Impacts of Cultural Factors on Multi-cultural Software Development Teams." 25th International Conference on Software Engineering, IEEE, 2003.
- ⁶¹ Nicholson, B., S. Sahay, and S. Krishna. "Work Practices and Local Improvisations within Global Software Teams: A Case Study of a U.K. Subsidiary in India." Proceedings of the IFIP, Working Group 9, 2000.
- ⁶² Walsham, G. "Globalization and ICTs: Working Across Cultures." Research Papers in Management Studies, The Judge Institute of Management Studies, Cambridge UK, 2001.
- ⁶³ Greenwood, T. G. "International Cultural Differences in Software." *Digital Technical Journal*. Vol. 5 No. 3, 1993
- ⁶⁴ Kersten, G., Kersten, M., and W. Rakowski. "Application Software and Culture: Beyond the Surface of Software Interface." InterNeg Reports INR1/01, 2001.
- ⁶⁵ Jackson, L. A., von Eye, A., Biocca, F., Barbatsis, G., Fitzgerald, H. E., & Zhao, Y. (2003). Internet attitudes and Internet use: Some surprising findings from the HomeNetToo project. *International Journal of Human Computer Studies*, 59(3), 355-382.
- ⁶⁶ Sun, H. "Building a Culturally Competant Corporate Web Site: An Exploratory Study of Cultural Markers in Multilingual Web Design." Proceedings of the SIGDOC, ACM, 2001.

-
- ⁶⁷ Srinivasan, V. "O Outsourcing and Offshoring" Presentation to MIT Special Seminar on International Management - Offshoring, Feb. 25, 2004
- ⁶⁸ Saini, Sanjay. "Offshoring Experiment in Radiology". Presentation to MIT Special Seminar on International Management - Offshoring, Feb. 25, 2004.
- ⁶⁹ Suh, Bob. "Perspectives on Global Delivery". Presentation to MIT Special Seminar on International Management - Offshoring, Feb. 25, 2004
- ⁷⁰ Malhotra, Raj. "Unique Business Model." Presentation to MIT Special Seminar on International Management, Apr. 10, 2004.
- ⁷¹ Ellis, John. "Building an offshore contract engineering business." Presentation to MIT Special Seminar on International Management – Offshoring, Apr. 10, 2004.
- ⁷² Shah, Swapnil. "Offshore Development." Presentation to Massachusetts Software Council, Apr. 13, 2004.
- ⁷³ Swadia, Sandeep. "Increasing Software Development Efficiencies." Presentation to Massachusetts Software Council, Apr. 13, 2004.
- ⁷⁴ Baxter, Stephan P. "A Business Primer: How to Expand into China". Presentation to MIT Special Seminar on International Management - Offshoring, Apr. 10, 2004.
- ⁷⁵ Andre, Dave. "Case Study: Technology Offshoring at Upromise". Presentation to Massachusetts Software Council, Apr. 13, 2004.
- ⁷⁶ Sukharev, Alexis. "Auriga". Presentation to Massachusetts Software Council, Apr. 13, 2004.
- ⁷⁷ Barley, Stephen. "Technology as an occasion for structuring: Evidence from observations of CT scanners and the social order of radiology departments." *Administrative Science Quarterly*, Vol. 31, 1986.
- ⁷⁸ Tuckman, B. "Developmental Sequence in Small Groups." *Psychological Bulletin*, Vol. 63, 1965.
- ⁷⁹ "National Innovation Initiative." *U.S. Council on Competitiveness*. <http://www.compete.org/nii>
- ⁸⁰ Wiederhold, Gio: "Outsourcing as Export of Intellectual Property"; *KD (Knowledge Discovery) Nuggets*, Vol. 10, March 2004.