Toward the 24-Hour Knowledge Factory in Software Development

Satwik Seshasai (IBM and MIT)

Amar Gupta (University of Arizona and MIT)

Introduction

"The Sun never sets on the British Empire," was a notion emphasized during the eighteenth and nineteenth centuries to highlight that the British Empire was far-flung, and that the sun was always visible from some part of this vast empire. While the British Empire has gradually disintegrated, we can now coin an equivalent notion: "The Sun never sets on the 24-hour Knowledge Factory!"

The notion of the 24-hour Knowledge Factory can be traced back to the industrial revolution. Since the installed equipment was scarce and costly, the employees were scheduled to work in shifts of 8-16 hours in order to use the manufacturing facilities on a round-the-clock basis. With the advent of electronic computers and diminishing costs for telecommunications, one developed the notion of 24-hour Call Centers. Depending on the time of the call, it is automatically directed to a call center that is active at that time. Using a cluster of 3 to 4 call centers located in time zones 6-8 hours apart from the time zone of the neighboring call center, one can ensure that all employees of these geographically distributed call centers are working during daytime in their respective countries. The notion of multiple support centers was subsequently adapted for supporting global communications networks over time. Now it has become feasible for one to use a geographically distributed workforce of highly trained professionals to complete an endeavor in a much shorter timeframe as compared to a scenario in which all personnel are based at one location, irrespective of where location is.

By involving specialized microchip design engineers located at multiple places around the world, a semiconductor chip design firm may create virtual "24-hour knowledge factories". This allows for an efficient design process that has a faster turnaround time. It provides the firm with access to high-talent designers who would otherwise have to move to a different country, or work at odd hours of the night; some persons call the latter type of shift as the "graveyard shift". The creation of professional service teams that transcend geographic and temporal boundaries offers the potential to change the face of many industries. This innovation will dramatically

impact the manner in which companies build, test, sell and support their products and services. Years ago, people in India and the United States thought the time difference was a negative - they thought it would hinder their ability to work with US firms. Now that has switched around - for many projects the time difference is a plus, as it enables the creation of the 24-hour Knowledge Factories described in this paper.

The 24-hour Knowledge Factory will involve "offshoring" of part of the endeavor. Today, offshoring is done primarily to reduce costs. We believe that over time, the growth in offshoring will also be fueled by the potential to achieve drastic reductions in turnaround times for major endeavors. The focus of this paper is on software development, and specifically on new product development. The authors contend that efficient information management is the key to incorporating 24-Hour Knowledge Factory concepts in such development efforts, and describe models for achieving strategic advantage with work teams located in three continents of the world.

This paper uses a case study to highlight a 24-Hour Knowledge Factory model with integrated data analysis. While this study involved two sites within IBM, the findings and methods could be applied to endeavors that use three or more geographically dispersed sites within any corporation or across multiple collaborating companies. As compared to traditional single-site operations, significant differences were observed in information sharing, collaboration, and innovations in work operations. The quantitative measures used in this case study gauged data on aspects such as frequency and methods of collaboration, social and technical networks, and differences in handling strategic and tactical decisions. The qualitative parameters were elicited through interviews in which the stakeholders described their perceptions of the quantitative data, and their motivations for decisions related to knowledge sharing. The primary emphasis of the field study was to evaluate the role that spatial and temporal differences play in the creation of new software products, with this analysis serving as the foundation for studying the 24-Hour Knowledge Factory paradigm.

Defining the 24-Hour Knowledge Factory

The term "24-Hour Knowledge Factory" connotes a global delivery model in which members of a global team work on a project around the clock; each member of the team works the normal workday hours that pertain to his or her time zone. At the end of such a workday, a

fellow team member located in a different time zone continues the same task. This creates the shift-style workforce that was originally conceived in the manufacturing sector. A globally distributed 24-hour call center is the simplest manifestation of this paradigm. A better example of the 24-hour factory paradigm involves groups working together to accomplish a given set of deliverables, such as a software project, and transcending conventional spatial and temporal boundaries.

Software development involves the creation of a product that is produced primarily through the transmission of knowledge between members of the development team. The figure below illustrates a distributed factory with software design operations in three countries around the world. In this particular delivery model, each geographic location is responsible for a separate task, and the overall efficiency of the project is improved since each location perceives that progress is made "overnight" when workers at that location are asleep. Additional models, discussed in the following sections, are characterized by different distributions of tasks depending on the appropriate needs for information management.

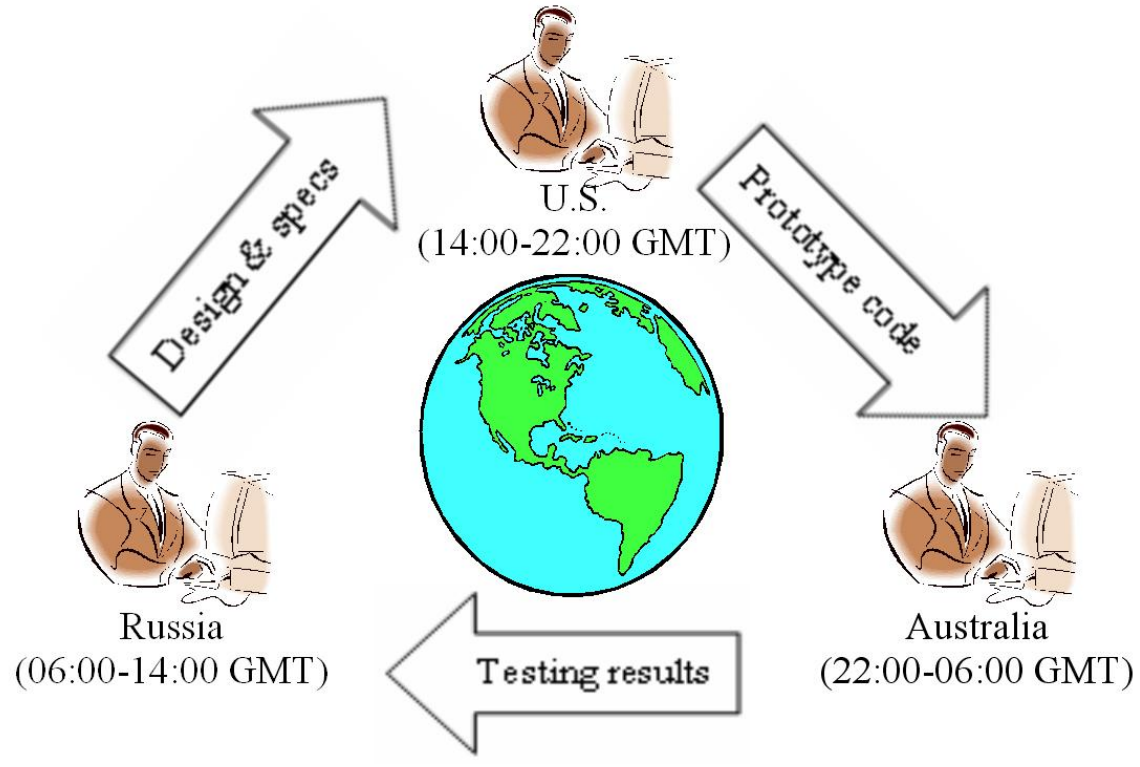


Figure 1: A 24-hour tri-foci scenario.

In a "24-hour software development environment" that encompasses three or more development centers located around the world, the distributed team is envisaged to concentrate

on the same problem and to perform the same function (whether it be development of code or testing of subsystem) on a successive basis, with each collaborating center retaining ownership of the endeavor for 8-hour periods in every 24-hour cycle. Many industries, including the software industry, are characterized by a development cycle that relies heavily on sequential performance of specific functions, such as development, testing, and verification. In a traditional software development environment where all parties are located in the same geographic area, a code developer typically waits until a fully functional portion of the product is available before passing it on to an engineer to test it. However, with the potential for receiving testing feedback overnight, the developer now has the unprecedented opportunity to build portions of the product on an incremental basis.

Foundations of the 24-Hour Knowledge Factory

The 24-Hour Knowledge Factory model can benefit from existing models for plant location, primarily in the manufacturing industry. These models were combined with frameworks for organizing decision making structures in a global organization and brought into the domain of knowledge based products and services. By examining the factors that caused plants to be located in various geographic locations, one can begin to determine appropriate decision criteria for producing knowledge based products such as software at various geographic locations. Once the geographical decisions are made, the next important step is to incorporate decision-making and knowledge management structures for geographically and temporally distributed locations, so that the "plants" can operate in a seamless and productive supply chain.

A number of researchers have looked at various issues of relevance to the 24-Hour Knowledge Factory. However, the work in this area has generally neglected to treat the global delivery system as a knowledge factory, where knowledge is the key component that is produced and traded– this is the innovation that this paper is emphasizing. Software is being used as a prime example of this model, but the model itself applies to any domain in which knowledge is the core component of production. In constructing this 24-Hour Knowledge Factory model, a number of known concepts can provide valuable building blocks. For example, prior research on optimal plant location for global manufacturing can be applied to the knowledge manufacturing domain. Similarly, ongoing research on software management practices can be adapted for use in a globally and temporally distributed framework. Finally, the growing notion of sourcing

work offshore provides the foundations for discussing our "hybrid" model – where work is shared between onshore and offshore locations, rather than sourced completely. This paper builds on the existing body of literature on offshoring by considering the time difference between sourced locations to be a benefit, rather than a barrier to overcome, and by considering tasks shared between locations, rather than be assigned from one location to the other.

Global Manufacturing - Plant Location: Lovelock points out that lessons from the manufacturing arena can be applied to services, by understanding global drivers such as location choice[1]. 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 plants[2]. 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 interfaces[3]. Locations should be seen as value centers, with the specific source of value for the particular center being explicitly identified[4]. 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 the entire lifecycle of the product[5]. Thus, in building a global knowledge factory, companies cannot just cut overhead costs as they would in a typical offshore environment; instead, they need to transform manufacturing processes for long-term success[6]. Pisano points out that knowledge-based companies cannot just invest in innovative R&D and outsource manufacturing – they need to invest in manufacturing process too[7]. 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.

Offshore Outsourcing: Agrawal et. al. describe "round-the-clock shifts" offshore, where some firms even pay higher wages for offshore workers to work odd hours[8]. Their research has found that companies can reduce costs by 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 ventures[9]. This framework is useful in creating the hypothesis that the 24-hour model is the next natural step in this progression.

Many researchers have described emerging facets of offshore outsourcing that motivate the 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 capability[10]. Barney cites transaction cost economics as the only factor in determining whether to keep tasks within company boundaries[11]. Carr suggests that offshore models cannot just modularize tasks, and emphasizes the need to build strategic competencies in all locations for all tasks[12]. Three factors always exist for competitive advantage, according to Christiansen - economies of scale and scope, integration and non-integration, and process-based core competencies[13]. Light indicates that managers must understand cultural values of themselves and their employees in all locations to be successful[14]. DiRomualdo and Gurbaxani provide more guidelines for assessing outsourcing: improvements to information systems, business impact, and generation of new revenue[15]. In placing the offshore outsourcing thrust in a national context, Young states that the U.S. must try harder to compete globally, because individuals in the U.S. cannot give up their standard of living[16].

As the 24-hour knowledge factory concept progresses, 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 organizations[17]. Champy describes X-engineering, a new model that is changing organizational relationships by introducing transparency, standardization, and harmonization[18]. Organizational learning is cited as one major factor in building flexible offshore models[19]. Lacity et. al. agree that organizations should focus on continuous learning more than the strategic offshoring versus commodity debate[20]. Furthermore, they state that firms should concentrate on efficiency, not costs, when choosing an outsourcing location[21], and the maturity of the technology should be part of the decision on what to outsource[22]. 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 maximum[23]. Fuchs notes the need to align product-market focus, resources and capabilities, organizational culture, and direction – this is a challenge to be faced with many global centers, especially when working on the same task, as one virtual organization[24].

The GLOBE study highlighted that cross-cultural teams need to be managed with an appreciation for the specific cultural drivers that exist in each individual culture[25]. Emerging markets need to be treated differently from 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 change[26].

The multi-shoring concept is offshoring to multiple locations, based on skills availability and competencies[27]. 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 processes[28], a motivation for the 24-hour model proposed here which is not possible without outsourcing. Cheifetz advises companies to outsource strategically, and to beware of cultural differences[29].

With regard to the value chain and decisions on what is best to outsource, researchers have come up with conflicting advice. Chesbrough states that outsourcing innovation would involve negative ramifications when done across countries because of inherent conflicts of interest[30]. However, Quinn states that outsourcing innovation is one of the means for companies to succeed – he suggests that outsourcing needs to take advantage of talent at all stages of the value chain; accordingly, new product development can be done in offshore locations[31].

Organizations need to understand the "ecosystem" within which they exist, and to consider the health of the organizations around them which affect their performance[32]- 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 decisions[33].

Kumra states the need to distinguish between activities that require proximity to the customer and activities that can be done remotely, and to distribute tasks appropriately[34] – in our

model, a given task can have a customer-facing component as well as a remote component and one can take advantage of both since the tasks are shared between locations. In this complex environment, relationship management is the key to outsourcing ventures, because the situation almost always changes from what was initially planned[35].

Finally, outsourcing requires management styles that involve negotiating results rather than issuing orders[36]. In a 24-hour knowledge factory, this notion needs to be taken one step further: the factory should allow the employees in each location to negotiate tasks with each other, since the degree of coordination required does not allow for a centrally managed system.

Information Technology Management: Taylor and Bain treat the call center as a "white collar factory", and incorporate employment relations concepts from the factory[37]. The 24-hour knowledge factory proposed 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 itself 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 success[38]. Aron and Singh note that IT work is on a knowledge continuum – and information workers play a key role at each stage[39]. When managing a software team, management practices in America cannot always be transferred to other countries, so one needs to understand the cultural values involved as well[40]. 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 products[41].

Knowledge Management: Barthelmy suggests that managing the sourced effort is the largest hidden cost – and organizations should keep their core knowledge within the core of the company[42]. Lei and Slocum, on the other hand, argue that alliances between sites must understand and share core competencies, not be "deskilled"[43]. Tallman addresses the need to take advantage of knowledge networks around the world using capability based theory where the process can act as leverage[44]. Powell states that knowledge-creation is the primary core competency for firms, and collaboration is vital to this creation of knowledge[45]. Continuing this

point, Davenport notes that managing customer support knowledge is vital – it is impossible to replace this direct, human input with automated tasks[46]. Companies that succeed will have methods for managing the knowledge within their firms to ensure that duplication of effort is eliminated[47]. 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 to share it the most.

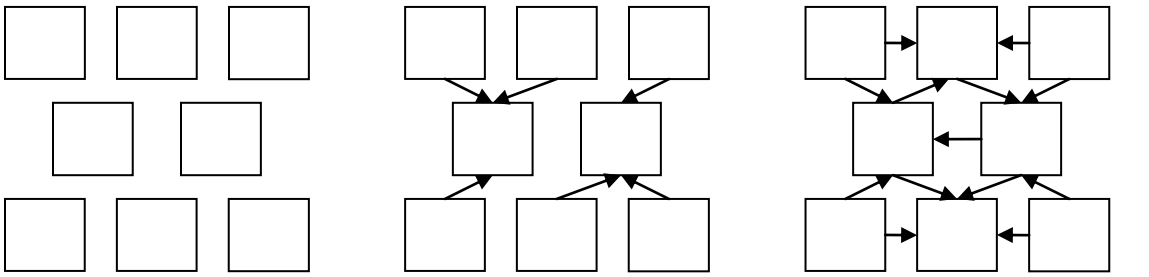
New Product Development: In new product development, researchers have identifies several factors to be relevant to a successful global delivery model. Coordination between locations is key (not decentralization), and configurations are different for firms such as Boeing and others who use multi-national models for new product development[48]. Earl presents risks for distributed new product development such as the loss of organizational learning and innovation[49]. 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 groups[50]. At Dell, technology was used to innovate the coordination activity between different parts of the supply chain so that all organizations were treated as one company[51]. 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 that drives the 24-hour knowledge factory to maintain "plants" in various locations, but to share tasks horizontally between locations.

Modeling the 24-Hour Knowledge Factory

The salient characteristics of the 24-Hour Knowledge Factory paradigm can be understood through a series of inter-related models. The first is a taxonomy for modeling tasks within the software domain. The next is a taxonomy for modeling organizational hierarchies that could be employed in building a 24-Hour Knowledge Factory. These axes provide the means for assessing the knowledge management needs of the particular application or operating environment, and also for managing the complex information flows within the organization. The information flow within a 24-Hour Knowledge Factory can be configured as a set of decisions –

decisions regarding which projects to undertake, which people to consult on which projects, and which technical decisions to make with regard to all the stakeholders involved.

Taxonomy for Task Dependencies: The three scenarios depicted in Figure 2 illustrate the situations that may apply to a distributed software support center, a software maintenance engineering team, and a new product development environment respectively.



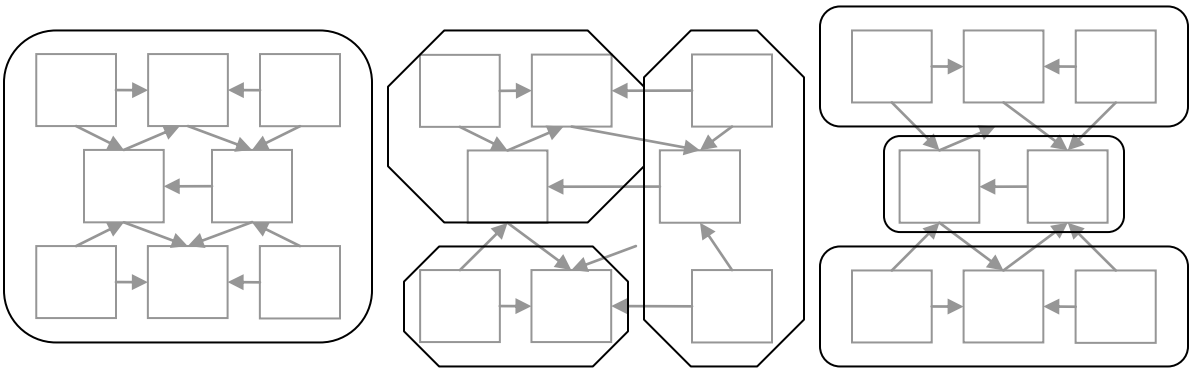
(a) Autonomous **(b) Semi-Autonomous** **(c) Tightly-interdependent**

Figure 2: Decision-making dependencies for individual work: three scenarios.

In the autonomous scenario (first case), individuals work relatively independently and do not rely on others for advice in making their decisions. An example of this is a software support center where customers can call an individual support representative and receive knowledge from that one individual. In the semi-autonomous scenario (second case), individuals still work independently, but occasionally need to consult others with more expertise, creating a hierarchy. An example of this is a software maintenance engineering team assigned to develop incremental releases to an existing software product. Such a team can work somewhat independently because the changes to the code are primarily isolated bug fixes, but may occasionally need to consult experts such as the original developers of the code. The third scenario involves individuals who are heavily interdependent, such as members of software new product development team; the decisions made by one team member have impact on many other team members, and also require the inputs of many other team members.

Taxonomy for Organizational Hierarchies: Another axis to consider while considering appropriate decision support systems is the nature of the organization. In a flat organization, all decision-makers, regardless of task or geography, can be deemed to belong to a single organization. In other cases, additional layers of hierarchy exist within the overall organization.

Based on the degree of importance played by the geography or the task, one can visualize the three cases, depicted in figure 3, to filter decision making input and output through either the geographic location or the task group.



(a) Flat organization **(b) Geography Specific** **(c) Task Specific**

Figure 3: Organizational models for heavily-interdependent decision making teams.

Although the flat organization may seem simple, it is the most complex model from the viewpoint of the 24-Hour Knowledge Factory because individuals must consult with the maximum number of other individuals without the benefit of levels of hierarchy to aggregate the various inputs and outputs of their decisions. Consider the scenario where designers of the software system are located in Boston, China and Germany; and testers of the system are located in China and Australia. The stakeholders for the system may be located in other countries and time zones. Designers in Boston may need to consult testers in China for performance analysis, who may in turn need to consult stakeholders in France for performance measures. This complex decision-making scenario is more difficult to handle with a peer to peer approach in a flat organization model, as compared to organization models in which some notion of hierarchy has been pre-established based on geographical or other considerations.

Information Management Framework for 24-Hour Knowledge Factory

Virtually all possible types of software development endeavors can be categorized using the couple of taxonomies discussed above. Many of these endeavors involve rapid and coordinated decision making. In order for these decisions to be made in a timely and effective manner, one needs an integrated decision support systems to mitigate the type of diverse problems encountered with a geographically and temporally distributed decision making team. In

the 24-hour Knowledge Factory, knowledge is passed from location to location in rapid motion, and must be characterized and structured appropriately to minimize the time required for a worker to absorb the previous knowledge worker's information.

A concept demonstration prototype, called KNOWFACT (from Knowledge Factory), was developed by the authors at MIT to demonstrate and extend potentially useful methodologies and technologies. This prototype was tested in a real-world situation involving the design of satellites, where technical decisions regarding the specific design of the system are made on a daily basis, and require constant re-evaluation of the effects of these decisions on a variety of stakeholders who were geographically distributed[52]. This type of model is very relevant to the programmers in order to attain common understanding of requirements and of past experiences and behavior; such knowledge of historical facts and issues can dramatically enhance the productivity of workers involved in development, testing, documentation operations in the 24-hour software knowledge factory.

The KNOWFACT decision support system is depicted in figure 4. In this diagram, the Decision Rationale Module (DRM) facilitates the definition of the key attributes that characterize the system for each stakeholder; these attributes form the basis of a utility interview which helps determine the level to which the present state of the system satisfies the stakeholders (or requirements). The value of this system to the manager is that it provides a structured and consolidated system for forcing the team to represent only the most important factors which drive the software system being produced. The Decision History Module (DHM) captures all the historical information on specific decision parameters; and the values for these parameters are aggregated to calculate values for the attributes which were defined for the DRM. The system was designed to incorporate only a small set of aggregate attributes based on earlier research that decomposing a system into a small set of aggregate attributes has a positive effect on human comprehensibility and accuracy in terms of interacting with a decision support system[53]. Dynamic forms are used for human interaction with the system, because decision support systems with flexible forms that are appropriate to the data being used for decision-making are most useful to decision-makers[54]. As the final step of the system, the calculated values for each attribute are placed into the utility function to calculate the overall utility measure for the system; this utility measure is used by the team to redefine which attributes to use, and to store the rationale for the decision.

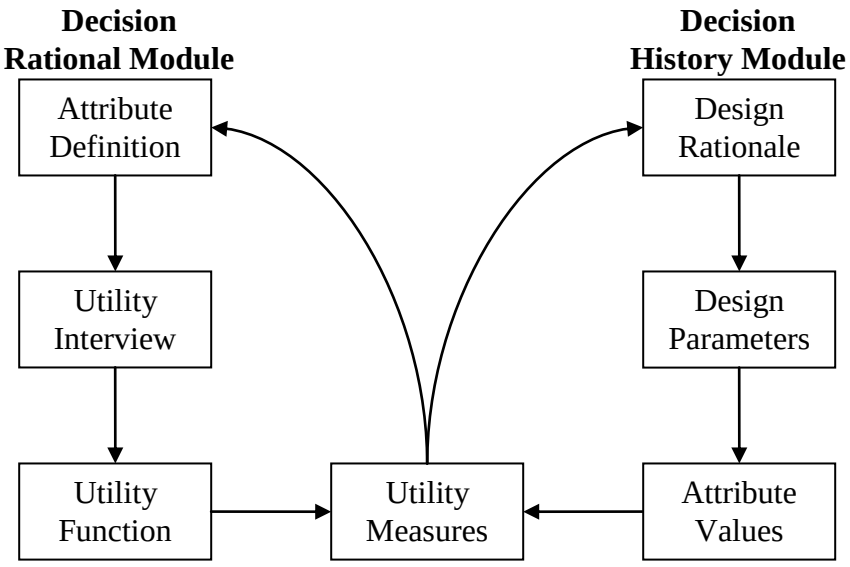


Figure 4: Architecture diagram for KNOWFACT paradigm

DHM provides the capability to perform automatic capture of vital information as the decision process evolves over time. It allows the user to view all the information in a graphical format, records the history of the states of the concerned variables, performs integrity checks and facilitates the capturing key elements. In addition, it provides a centralized repository of relevant information. DHM facilitates the creation of an evolving knowledge repository that contains knowledge on the current state of the activity being undertaken, the history of various states of the entities, and an automatic analysis of system integration integrity; the latter is performed with the objective of alerting the integrator of potential conflicts. The forms used to capture decision rationale input build on decision support research into the capabilities and structures of forms to structure input forms which recognize major changes in the design and prompt for rationale input only at the crucial moments.

The DHM system is geared to capture, reuse, and better exploit valuable information assets, with the objective of mitigating temporal and spatial barriers in large multi-organizational multidisciplinary endeavors. In a number of applications ranging from marketing campaigns of new products to the design of new systems, each campaign or design process is frequently conducted as a new endeavor. Very little knowledge, if any, is carried over from the previous episode or campaign. In order to mitigate this problem, a new approach was developed to provide automated capture and storage of important information (name, value, rationale, author, etc) relating to each component of the project, along with the history of these data. While most of this functionality is automated, the user is prompted to document the rationale when

significant changes are made to the design or the implementation. This concept demonstration system exemplifies how the 24-hour knowledge factory can allow projects to be transferred easily between teams, without requiring each team to invest time in transferring the knowledge of previous projects; it can also facilitate teams to look at decisions made on earlier projects that involved decisions on similar issues.

The Decision Rationale Module (DRM) is geared to elicit and capture critical information on the objective of the endeavor, and to analyze and store information about utility characteristics for every stakeholder involved in the particular endeavor. This builds on prior research that suggests that decisions across teams are best made when the decision can be characterized in terms of multi-attribute models and where each attribute represents an aggregated set of characteristics of the system. The DRM system builds on this model by allowing stakeholders to interact at the attribute level, rather than having to use the attributes simply to break down the system into specific parameters. The specific parameters involved in the decision are related to the attributes once; and then the utility measures are used to draw exact links between the parameter changes and the effect on the overall attributes.

The evaluation of the utility function, through the definition of attributes for every stakeholder, allows one to create, define and analyze attribute information, to structure the knowledge and to build a series of rule-based interviews with the objective of eliciting the stake holder's utility function. These interviews build upon earlier research on form structures to build the interview form. The interview questions are auto-generated based on existing templates and the data provided by the designers about the appropriate ranges for each attribute. These forms are dynamic, building on previous research that shows that dynamic forms lead to a higher quality of output. Additionally, the look and feel of the interview form has been designed to allow the users to visualize the utility implications of the input they are providing.

DRM allows users to observe links between multiple stakeholders, multiple decision phases and multiple projects. It provides a formalized means of exploring a trade space by incorporating preferences into decision criteria with methods based in economic and operations research theory[55]. Both utility analysis methods and cost-benefit analysis methods are employed to obtain information from all of the stakeholders in the geographically and organizationally decentralized design process and to facilitate communication between them. The principal mechanism for eliciting information is by conducting a series of interviews with the stakeholders;

these interviews are domain-specific and are designed to help capture the users' preferences regarding the various attributes of the design architecture. These data are then used to drive the decision process, by providing information about the utility and cost of each potential architectural alternative. In the 24-hour knowledge factory environment, such a system would allow for the opinions of the decision-makers' to be adequately represented even when they are not available at the same time or place.

System Dynamics approach to explore potential barriers to 24-Hour Knowledge Factory

A Systems Dynamics approach can be employed to assess 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. The model here is not exhaustive – instead, it can be utilized as the foundation for incorporating additional factors and dependencies. Several of the factors shown on this figure emerged from presentations by guest speakers in the course on offshoring conducted at MIT and at the University of Arizona, as well as and a symposium organized by the Massachusetts Software Council.

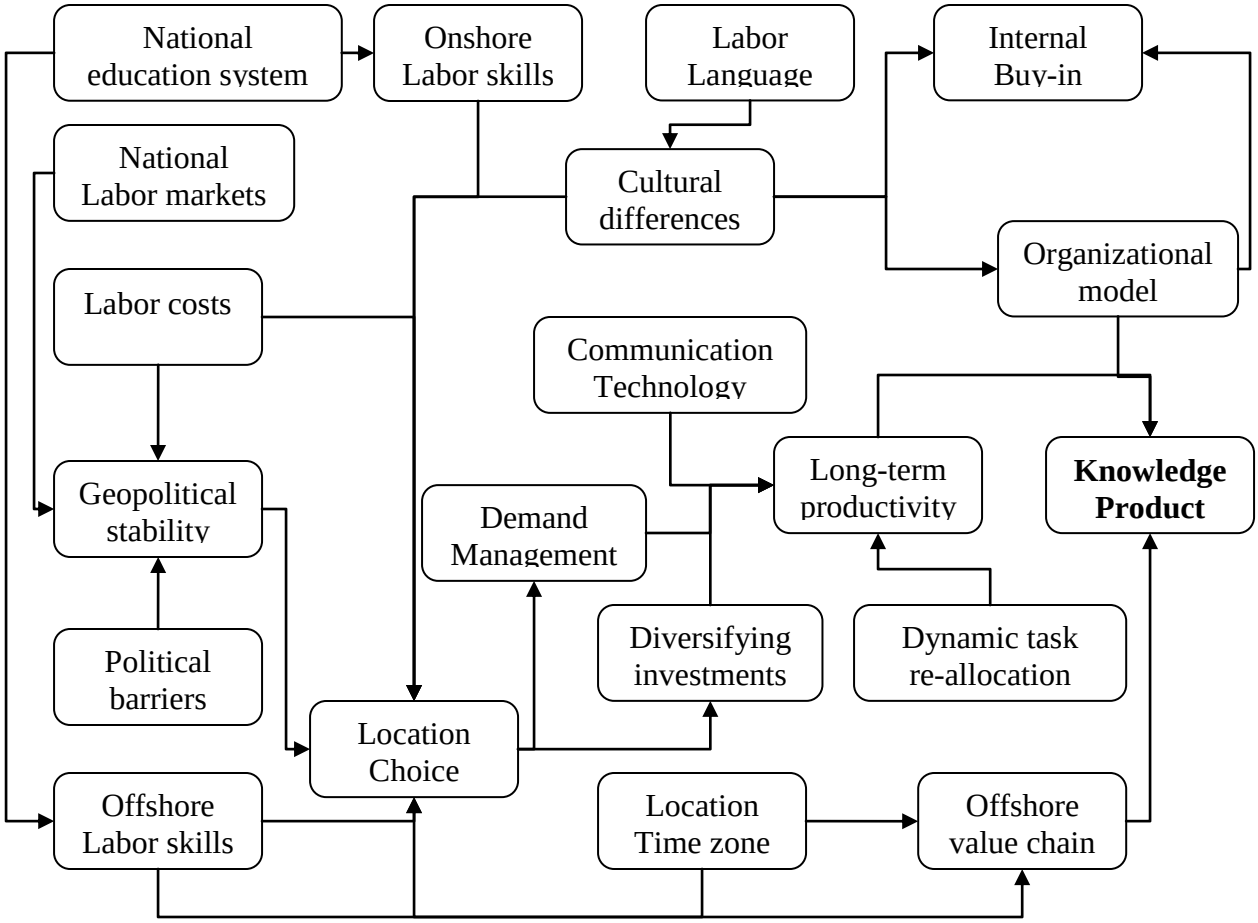


Figure 5. Systems model for 24-Hour Knowledge Factory.

The system dynamics model highlights the complexity of the system, and the underlying interdependencies. The issues, delineated in the above figure, emerge in firms of all 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. Based on their broad relevance and significant potential impact, we explain the terms used in the above figure.

Demand Management: A 24x7 development model allows for much greater management of customer demand. This model can promote a faster time to market for products, and can 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 have[56]. 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 limited[57]. 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 resources[58]. In a company that employs home-based workers in India to perform medical transcription, the home-based work environment enables workers 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-time[59].

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 provide[60].

Long-term Productivity: The use of the 24-hour knowledge factory model 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; however, in India, a highly skilled radiologist will see employment by a U.S. hospital as a high-value position regardless of the task[12]. 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, 61].

The key here is to "transform, not transfer" the work[13]. The 24 hour knowledge factory is a different paradigm for knowledge-based work, and involves 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 adapt[15].

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-run[62].

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. 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% of the work 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 projects[13]. 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 offshore[12]. 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 chain[63]. 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 transition into high-value services, such as optics design, at a lower cost to outsourcing firms[15]. 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 [16,64].

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 required[13]. 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 size[11]. These axes need to 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 model[15].

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 apply[17].

Common Barriers within Firms: 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 project[11]. 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, firms are typically advised to 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 decisions[19].

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 pool[18]; and Russia has a legacy of USSR investments in science and a solid brand that has not been exploited to the level of India[15,65]. 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].

Current State Analysis – Case Description

Currently, few firms use the global delivery model described in this paper for purposes of new software development[66]. Some firms have adopted a similar version, with workers at two geographic locations (rather than three).

Mukherji and Ganguly cite experiences with offshoring simple and complex software projects to various geographic locations[67]. In one example, they developed a tactical, arms-length relationship with a software vendor in India, and this relationship allowed them to utilize the Indian provider for relatively simple tasks only. In the context of our models discussed above, this example would suggest an "autonomous" task structure, with a "geographic" hierarchy, in which 24-hour development was not utilized, and thus the value of the Indian provider did not increase as time went by. In another example cited by Mukherji and Ganguly, a strategic relationship was built with a software team in Israel, whereby complex tasks were shared between the U.S. and Israel sites. In this example, the strategic relationship allowed for skills that were available at both geographic locations to be utilized simultaneously and tasks to be completed in a more efficient fashion. This example, in terms of the three organizational models discussed above, conforms to the "heavily interdependent" task structure, with a "task-specific" hierarchy. Since the emphasis was on the tasks, the knowledge was able to be transferred between the collaborating groups in the two countries. As the relationship between

the two groups progressed, the tasks performed by the Israel team moved higher on the value chain.

Ferdows cites examples of organizations that have invested in a more strategic relationship with an offshore unit to achieve much higher value from the offshore teams[68]. However, most of these strategic relationships involve a sharing of knowledge – the true objective of the 24-hour knowledge factory is to share not only the knowledge but the tasks as well.

One of the authors of this paper possesses significant experience working with a large multi-national firm with a significant offshore presence for software new product development. In one division of the firm, new product development is organized in the "task-specific" hierarchy discussed above, with small software development teams of 6-15 people each charged with the development of a specific product. The teams consist of members from various geographic locations and it is common for a task to be shared between multiple team members from different locations. In interviews conducted with 10 members of this team, located in the U.S. and India, the team has cited that having members located in various geographic locations has resulted in a variety of benefits, and a few downsides[69]. These pros and cons are discussed in the following paragraphs.

Diversification of Knowledge Resources: With any team, the first step in bringing on new members is transferring knowledge between team members. In the offshore model, as tasks are completed, knowledge is transferred and stored within the various global sites. In the autonomous task model, a specific piece of knowledge about a specific task is only held in the one location in which the task is completed, unless a method of knowledge dissemination is established. In the 24-hour knowledge factory, with tasks being interdependent, and shared between locations on a nightly basis, knowledge is disseminated as a natural part of the process, without any extra effort. The engineers interviewed cited this diversification of knowledge resources as being vital to their being able to assign any task to any location, on an as-needed basis. Thus, if a particular task had to be done as quickly as possible, the manager of the team could assign the task to the location that was entering its daytime hours. For example, if a bug was found in the software code at 5 pm in the U.S., the manager could assign the bug to be fixed by the Indian team without a need for significant knowledge transfer to take place. This

knowledge diversification will only exist in the 24-hour knowledge factory model, with tasks being shared by global team members.

Value chain movement: By engaging the offshore software team in all tasks, in daily communication with the onshore software team, the tasks completed offshore have been able to move up the value chain much faster than if a contract-vendor relationship was used. The interviews described a progression of higher value tasks. The offshore teams were treated as new members of the team, and introduced into the cycle with the simpler tasks such as fixing problems in the software. However, as the team members became more knowledgeable, it was possible to move them up the value chain, without a significant investment in training or knowledge dissemination.

Time for resolution of tasks: The interviews confirmed that once the offshore team achieved a satisfactory level of knowledge, many software tasks were completed in a much shorter time with sharing between onshore and offshore team members. Team members adapted to a one to one pairing between onshore and offshore members, and this limited and consistent partnership made it possible to pass tasks nightly with minimal time investment in knowledge transfer. Thus, a task such as fixing a bug, which may take a U.S. developer 4 days to complete, could now be completed in 2 days, with a U.S. and Indian developer working concurrently. Such an improvement in time-to-resolve was important as it greatly reduced the time to market for the product, and allowed new features to be incorporated quickly into a software product.

Earlier reporting of issues: One of the major uncertainties in new product development of software is the untimely reporting of a bug, late in the product cycle, which can delay the release to market of the product. The software team which was interviewed used a testing team in China to work with developers in the U.S. to test pieces of the code overnight as the code is being developed. With the 24-hour model, a U.S. engineer was able to complete an incremental improvement to their piece of the code during the day, then pass it to a Chinese test engineer with test instructions, and then return the next day with results which helped focus the efforts of the U.S. engineer and expose issues much earlier than if testing had been done later.

Unintended process improvements: The software team cited numerous process improvements brought about by the need to share information with a geographically and temporally distributed team. Databases were used to track information such as design decisions, review comments, and testing results. The team members agreed that when the team consisted of co-located team members working at the same time, much of the knowledge was distributed informally, and thus the knowledge capture component appeared to be a significant additional burden. With the introduction of the 24-hour development model, the decision rationale and history capture methods (discussed in this paper) became a natural part of the process, and the natural tendency of software engineers to avoid the knowledge dissemination tasks was overcome.

Cultural understanding incorporated within software: The software team cited improvements made to the software through team members' intimate knowledge of their own cultures, and the usage patterns of their own cultures. For example, a Chinese tester was able to point out that Chinese users sitting in cubicles would not tolerate a "beep" sound when they make an error while using the software, since in the Chinese culture, users may have more of an interest in "saving face"[70]. Another engineer described a situation where Japanese developers were able to help adapt to building a user interface which searched for users by titles, rather than names, which was a requirement for selling into Japanese enterprises.

The 24-hour aspect of the development process is pivotal here because the developers from different cultures are concurrently engaged on all tasks, and can provide their cultural input into all tasks at all stages of the process. Although software firms have many mechanisms for understanding requirements from different cultures, there is no substitute for the engineer building the system from the start with the cultural knowledge incorporated.

Downsides: The first major downside was the loss of informal communication. Much of the software development process involves informal design meetings, and reviews of design decisions. The engineers interviewed stated that often they would engage in an informal discussion with a fellow engineer which would lead to an unintended exposition of a major piece of knowledge which would have a significant impact on the project. It was often the case that engineers were not immediately aware of the task history of their fellow engineers, and such informal communication would lead to a realization that the knowledge required to resolve a

current issue could be provided by a fellow engineer. The engineers interviewed consistently stated the use of communication technologies as a replacement for informal and in-person interaction removed a considerable amount of the knowledge exchange.

Overcoming this drawback, in the context of the taxonomies presented in this paper, would require a stronger understanding of the knowledge required in the information management framework and a better understanding of how tasks relate, in the context of the task dependency taxonomy. Once this understanding is developed, the hierarchy defined in the taxonomy for organizational hierarchies needs to facilitate the communication between the appropriate members of the process, so that the knowledge exchanges which are lost in the geographically model can be recovered.

Current State Analysis – 24-Hour Knowledge Factories in Action

The 24-Hour Knowledge Factory model is most prevalent in the software industry. The mobile industry firm, WDSGlobal, utilized Extreme Programming methodology in a globally distributed, round-the-clock software development project [71]. The programming team was distributed across three sites (US, UK, and Asia) and each site had joint ownership of the code. The team was distributed to meet their customer's regional needs, while sharing the same code base to avoid redundancy and maintenance costs. number of operational lessons were learned from this project. The first thing that was done by the team was to hire coaches to teach two of the sites the practices of extreme programming, whereby programmers contribute to the same lines of code in tandem. The study also found that there were slight cultural differences that caused some confusion between the developers. Another problem they ran into was technical in nature - the amount of time it took to download code from certain locations slowed progress at times – increasing network bandwidth between Singapore and the US as well as changing the technologies used to cache source control data were both cited as opportunities for improvement.

The WDSGlobal project demonstrated a set of enablers for the 24-Hour Knowledge Factory. The team met face to face at the outset to get to know and trust each other. Daily handoffs and explicit pairings within the team kept the trusting relationship going throughout the project. The team used VNC and video conferencing to meet face to face and share their work with each other. The daily knowledge transfer began as a summary of the day's work although evolved to discuss what the individual had learned and objectives they had before they left. The

project cited a key lesson learned as maintaining an equal sized team in each location or else the location with the largest team will take over the design. They also realized that priorities were changing too quickly so they determined that the managers could only reprioritize things once a week to make the team most productive.

Access and support for regional customers combined with a continuous engagement on the project served to make the WDSGlobal project a success. Using knowledge from the WDSGlobal project as a key example of 24-Hour Knowledge Factory success, an in-depth study was designed to collected detailed data from archival records within a global software team at IBM which also employs the 24-Hour model.

Global Software Teams at IBM

The two software development teams studied at IBM 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, and 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. The meetings of both teams ran in the same format 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 year, and periodic fix packs to customers.

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. The products are distinct but in the same 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. The two products each require the same skills and technology to build. While controlling for all aspects described above and treating the geographic distribution as the key differentiating factor, the data presented in this section portray the full range of contrasting aspects of each team's information sharing processes. All figures in this section relate to weekly output of the given data set for the year 2004. 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.

These data provide insight into the dissimilar ways in which two different teams at IBM use the same processes and technologies to accomplish their respective goals. The variance in the 52 weeks of data suggests that specific structural determinants could be used to study the relative performance and efficiency of distributed and co-located teams.

Using Electronic Mail for Asynchronous Discussion: Figure 6 displays the average number of contributors per e-mail thread for each of the two teams. The figure accounts for threads in the last week an e-mail was sent, so this figure 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. 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, such as out of the office notices. 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 e-mail discussion is that team members can think about their responses and provide more detailed input. When discussions reach a significant length, they are moved to a discussion forum database where responses can be better tracked and archived. 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 to discuss the design, post the results, and allow the Indian team to provide feedback later in the day 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 collocated framework. The manager of both teams stated that the discussion forum database is common to both teams; however the distributed team is more prone to use it.

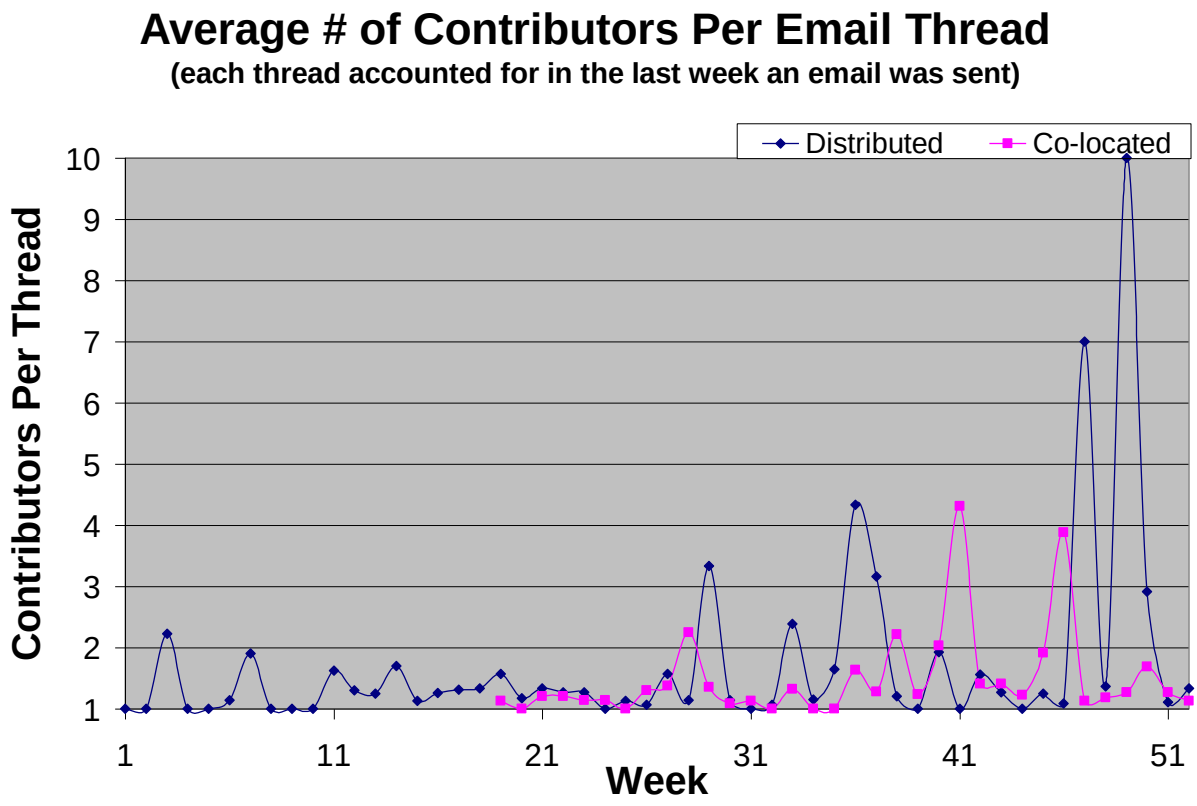


Figure 6. Average number of contributors per email thread.

The raw volume of email activity for the collocated team is higher (10.42 threads per week) than the distributed team (19.85). 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; however, on each of these threads, there is 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 substantially different.

Another aspect of these data 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 needed to be provided by the developers and sent to the testing team. After this week, the number of threads in the distributed team went down to an average of 8.91, while the number of threads in the collocated team went up, to an average of 31.45. A developer on the collocated team reacted to these data by noting that many problems are found by the testing team when features done by different developers interact with each other,

and the e-mail discussion is used to quickly remedy the gaps in knowledge between developers who worked on different 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.

Overall, the distributed team made much greater use of electronic mail (e-mail) as a forum for discussion, peaking during project deadlines. 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.

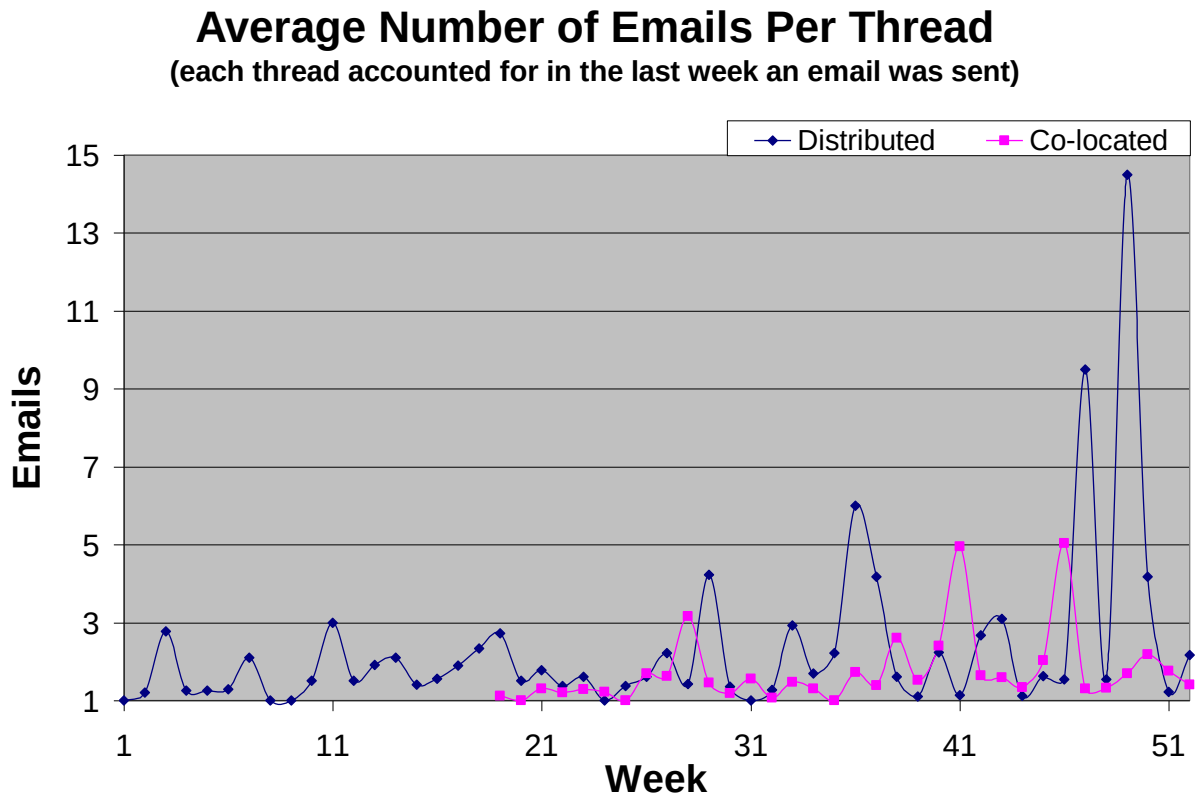


Figure 7. Average number of emails per thread.

Figure 7 highlights that longer e-mail discussions are the norm in the distributed team. Whereas Figure 6 addressed the number of individuals contributing to each thread, Figure 7 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. Overall e-mail

activity is greater in the collocated team, activity increases significantly after the feature freeze project milestone. The average number of emails in the distributed team was 17.06 while the average for the collocated team was 29.91. The fact that the collocated team's e-mail activity increased only after the project milestone is likely because decisions at this point were more tactical than strategic, and required quick resolution.

Technical Collaboration through Shared Source Code: Data from the source code for each team's product showed that the collocated team had a higher degree of technical collaboration and also had a higher degree of code modification after the feature freeze milestone. The weekly averages for source code modifications in the 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 steady state period up to the week 22 deadline, the average for the distributed team was 6.73 and the average for the collocated team was similar, 7.68. 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 speculated this was because questions were resolved face to face before 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. With the collocated team, the activity level rose after the feature freeze when code from individual developers was made to interact with code from the other developers. Figure 8 provides some insight into why this may have occurred.

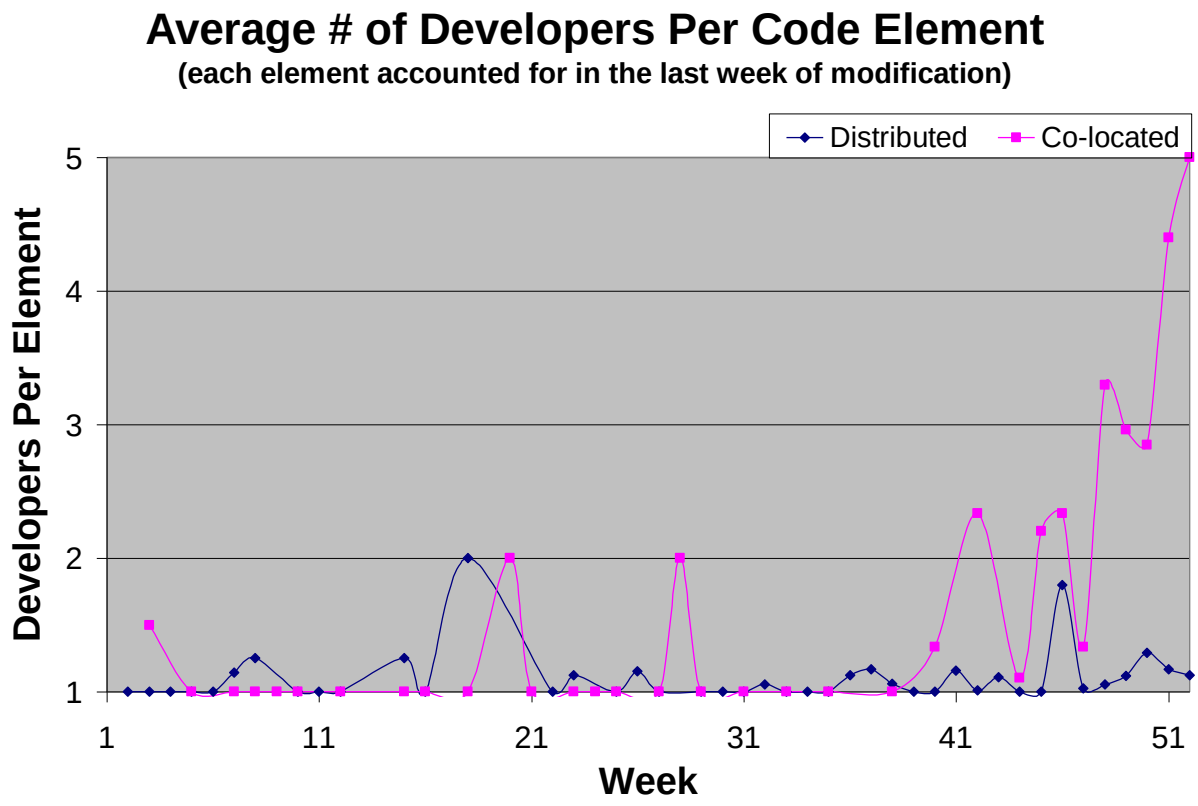


Figure 8. Average number of developers modifying individual source code elements.

Figure 8 displays the average number of developers per code element, with each element accounted for in the last week that it was modified. 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. Several members of the distributed team emphasized that they have drawn clear lines between code elements and that they try hard to modify only 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, except 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]." In contrast, while the collocated team does assign particular functional areas of the product to different developers, they will often reassign particular programming requests based on workload and feel comfortable with any developer modifying any part of the product. While the data do suggest more technical collaboration among the collocated team, there are code

elements on the distributed team which had 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.

Nature of Team Meetings: The team meetings held by the distributed team were found to be more tactical and task oriented than the ones of the collocated teams. Items were designated as being tactical if they were short term in nature and all knowledge related to completion of the item was already acquired – by this definition, 81.36% of the distributed team's items were tactical versus 39.25 for the collocated team. Examples of tactical items are issues related to the "build" (a compilation of source code into an intermediate internal product release to be sent for testing) and issues related to scheduling. Examples of strategic items are feature plans for the next release and cross-team collaboration with other teams in the company. Collocated Developer 4 stated that instant messaging or face to face office visits are used in lieu of meetings. On the other hand, Distributed U.S. Developer 3 stated that meetings provided the one chance for the team to be together and he often compiled a list of action items that he wanted to discuss.

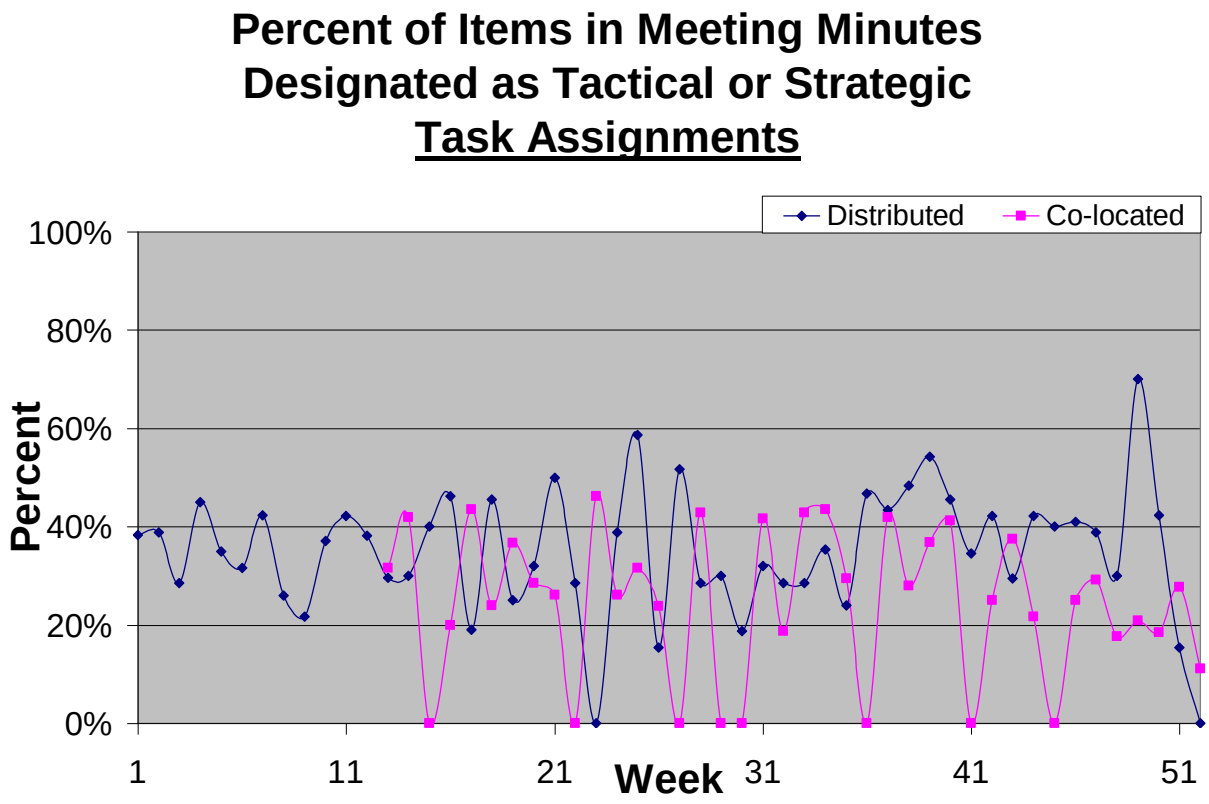


Figure 9. Percent of items in meeting minutes designated as task assignments (as opposed to status updates).

Figure 9 demonstrates two trends – the distributed team spends a larger percent of meeting items on "task assignments" (as opposed to "status requests") but both teams exhibit an alternating behavior of switching between task focused and status focused meetings. 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.

Using Technology to Update Work Item Status: Data from the Software Problem Report (SPR) database were useful in demonstrating how technology is used to update work item status. When any work is required on the source code an SPR is logged and is used to track the status. Despite this common purpose, each team was found to use the system differently. 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. 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. The average SPR actions for both teams were within 10% – the distributed team averaged 76.49 per week and the collocated team averaged 70.13.

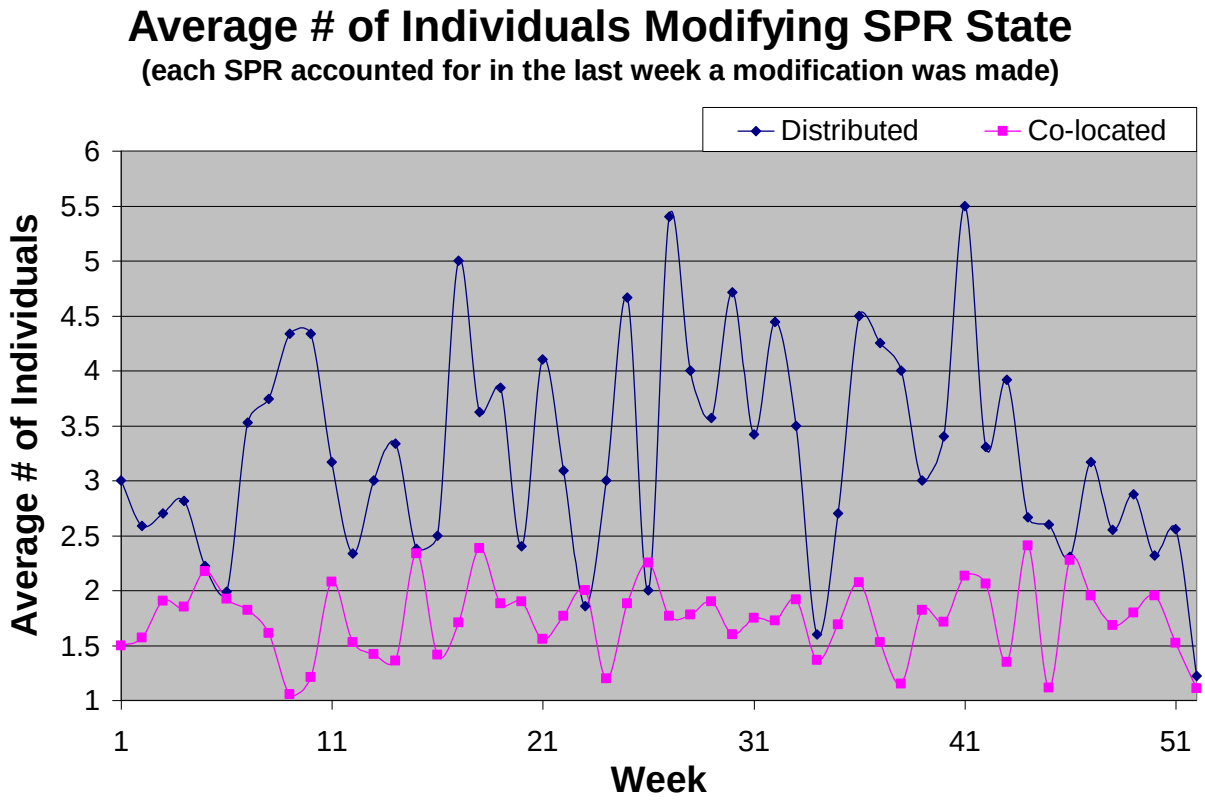


Figure 10. Average number of individuals modifying state of SPRs.

Figure 10 shows that the average number of individuals modifying the SPR state for a given SPR was higher for the distributed team (3.25) than the collocated team (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. When team members are not available immediately, it becomes useful to add updates to SPRs in the context of the particular issue and wait for a reply. With the collocated team, since answers are available immediately, it is not useful to take the time to update the formal SPR system. 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. The SPR system also provides a means for measuring some level of output from each team. The average time to resolution for the distributed team was 113.80 days while the collocated team averaged 120.72 days, suggesting that both teams were successful.

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. 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. The distributed team did not cite as great a use of informal communication, mostly due to the geographic separation and time zone differences. 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." 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.

Future State Analysis – Overcoming Barriers and Leveraging the Factory

The geographic structure of the teams at IBM that were studied led to different forms of value being achieved from their knowledge-sharing processes. Figure 11 highlights that the structure of the distributed team led it 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 should 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.

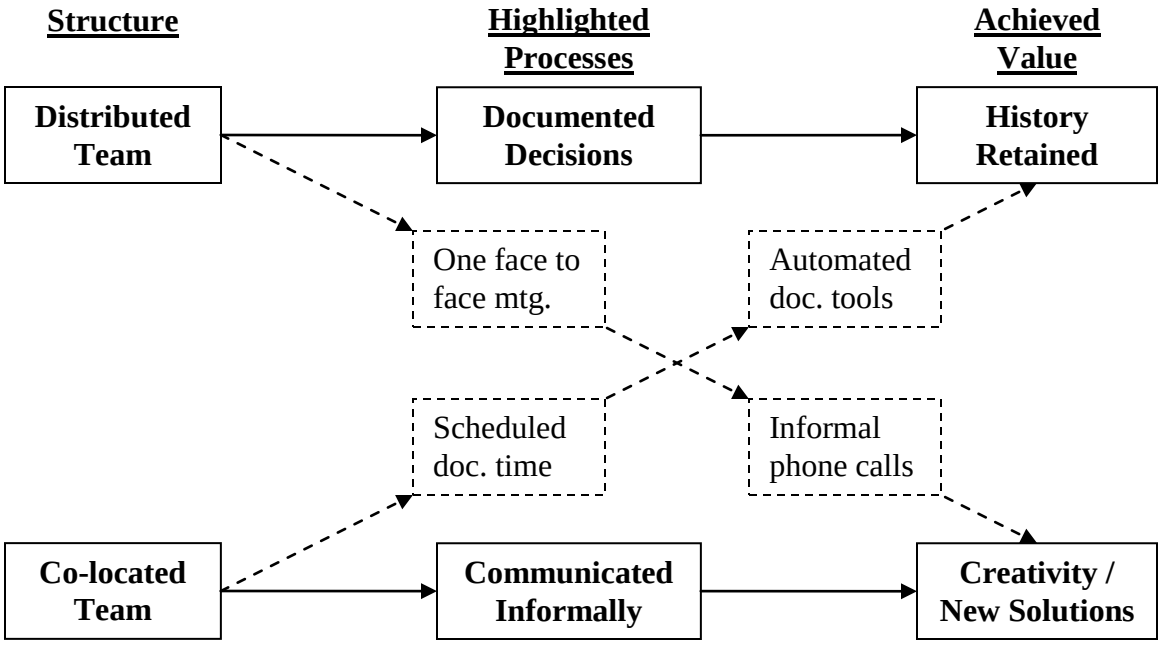


Figure 11. Scenarios demonstrating how to leverage a 24-Hour Knowledge Factory

Based in part on the IBM case study, the authors believe that the future of 24-hour knowledge factories is dependant on the reduction of barriers described in this paper, and a greater understanding of the potential benefits of such a model. For software firms to move towards this model, the following steps must be taken:

1. Assess the firm's software projects with respect to the task and hierarchy taxonomies presented above. By analyzing how interdependent the tasks and hierarchies are, one can determine how to move forward in developing a global delivery model that facilitates knowledge flow throughout the organization.

2. Assess the firm's decision rational and history system with respect to the information management framework. Determining whether the right information is properly acquired from the appropriate people and disseminated to the appropriate people – with as minimal burden as possible – will be vital to maintaining a 24-hour knowledge factory.

3. Redefine the systems uncovered in Steps 1 and 2 to take advantage of the 24-hour offshore model. The decision to build a 24-hour knowledge factory is not simply based on a

current state analysis of the firm's systems, but a dynamic redefinition of systems to match the global delivery model which can yield the best results for the firm.

4. Work towards reducing the barriers exposed in the system dynamics model which may prevent the smooth flow of knowledge required for the 24-hour knowledge factory. Such barriers may include communication technologies, economic and infrastructure support systems, training systems, organizational culture, and geopolitical risk management.

5. Build the 24-hour knowledge factory! Note that even if the firm is not ready today to enter into the 24-hour global delivery system, it is important to begin the process of investigating and assessing the firm's own characteristics with respect to the taxonomies we have provided – the key point is that the 24-hour knowledge factory can be seen as an ideal endpoint on the spectrum of interdependent tasks and organizations, but firms can place themselves at any point on the spectrum and still succeed.

Conclusions

The 24-hour knowledge factory model is an emerging model that can potentially bring the benefits of globalization to all parties involved. This paper has focused on the modeling and implementation of the 24-hour knowledge factory model. This model allows firms to integrate the contributions of key players from around the world and engage all contributors at all points on the value chain in the same tasks. Further, it provides a mechanism for U.S. engineers to maintain high-value input into tasks, while utilizing the cheaper labor of offshore partners to accomplish the lower-value tasks. This paper used the example of the software industry as the pioneer in the emergence of this model, because the software process is based purely on the transfer and creation of knowledge with minimal infrastructure requirements.

The parallel nature of the data from the IBM case study – 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 that can be leveraged, along with time zone differences. A number of benefits from leveraging the geographic structure were cited in the interviews with the distributed team. Example 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 methods of coding of archival data derived from e-mail, telephone, meeting and other interactions could also be used as a feedback tool that could be highly valuable for corporations seeking to quantify the impact of the 24-Hour Knowledge Factory. The evidence from this case study emphasizes that the spatial and temporal distributions in a 24-Hour Knowledge Factory should be looked upon as a characteristic to leverage rather than a barrier to overcome.

As this globally distributed work paradigm evolves, other industries will gradually adopt and benefit from the 24-hour knowledge factory model.

References

[1] Lovelock, C., Yip, G. "Developing global strategies for service businesses." *California Management Review*. Winter 1996, pp 64-86.

[2] MacCormack, A., Newman, L., Rosenfeld, D. "The New Dynamics of Global Manufacturing Site Location." *Sloan Management Review*. Summer 1994, pp 69-80.

[3] Bartmess, A., Cerny, K. "Building competitive advantage through a global network of competencies." *California Management Review*. Winter 1993, pp 78-103.

[4] Venkatraman, N. "Beyond outsourcing: Managing IT Resources as a Value Chain." *Sloan Management Review*. Spring 1997, pp 51-64.

[5] Seitz, M., Peattie, K. "Meeting the closed-loop challenge." *California Management Review*. Winter 2004, pp 74-89.

[6] Blaxill, M., Hout, T. "The fallacy of the overhead quick fix." *Harvard Business Review*. Jul-Aug 1991, pp 93-101..

[7] Pisano, G., Wheelwright, S. "The new logic of high tech R&D." *Harvard Business Review*. Sep-Oct 1995, pp 93-105.

[8] Agrawal, V., Farrell, D., Remes, J. "Offshoring and Beyond", *McKinsey Quarterly,* No. 4, 2003.

[9] Kaka, N. "A choice of models." *McKinsey Quarterly*. No. 4, 2003.

[10] Sanuders, C., Gebelt, M., Hu, Q. "Achieving success in IT outsourcing." *California Management Review*. Winter 1997, pp 63-79.

[11] Barney, J. "How a firms capabilities affect boundary decisions." *Sloan Management Review*. Spring 1999, pp 137-145.

[12] Carr, N. "In-praise-of-walls." *Sloan Management Review*. Spring 2004, pp 10-13.

[13] Christensen, C. "The past and future of competitive advantage." *Sloan Management Review*. Winter 2001, pp 105-109.

[14] Light, D. "Cross-cultural lessons in leadership." *Sloan Management Review*. Fall 2003, pp 5-6.

[15] DiRomualdo, A., Gurbaxani, V. "Strategic intent for IT outsourcing." *Sloan Management Review*. Summer 1998, pp 67-80.

[16] Young, J. "Global competition - the new reality." *California Management Review*. Spring 1985, pp 11-25.

[17] Mizoras, A. "In House versus Outsourced." *IDC Opinion*, IDC, 2004.
[18] Champy, J. "Is technology delivering on its Productivity Promise?" *Financial Executive*, October 2003, pp 34-39.
[19] McFarlan, F., Nolan, R. "How to manage an IT outsourcing alliance." *Sloan Management Review*. Winter 1995, pp 9-23.
[20] Lacity, M. Willcocks, D., Feeny, D. "IT outsourcing: maximize flexibility and control." *Harvard Business Review*. May-Jun 1995, pp 84-93.
[21] Lacity, M., Hirschheim, R. "The IS outsourcing bandwagon." *Sloan Management Review*. Fall 1993, pp 73-86.
[22] Lacity, M. Willcocks, D., Feeny, D. "The value of selective IT sourcing." *Sloan Management Review*. Spring 1996, pp 13-25.
[23] Quinn, J., Hilmer, S. "Strategic outsourcing." *Sloan Management Review*. Summer 1994, pp 43-55.
[24] Fuchs, P., Mifflin, K., Miller, D., Whitney, J. "Strategic integration: Competing in the Age of Capabilities." *California Management Review*. Spring 2000, pp 118-147.
[25] Light, D. "Cross-cultural lessons in leadership." *Sloan Management Review*. Fall 2003, pp 5-6.
[26] Arnold, D., Quelch, J. "New strategies in emerging markets." *Sloan Management Review*. Fall 1998, pp 7-20.
[27] offshoring drive.pdf
[28] Millman, G. "Going Beyond Commodity Outsourcing." *Financial Executive*. Sep 2003, pp 55-57.
[29] Cheifetz, I. "Think Like Buffet About How to Value Outsourcing." *Financial Executive*. Sep 2003, pp 58.
[30] Chesbrough, H., Teece, D. "Organizing for innovation." *Harvard Business Review*. *Best of Harvard Business Review 1996*, pp 127-135.
[31] Quinn, J. "Outsourcing innovation the new engine of growth." *Sloan Management Review*. Summer 2000, pp 14-29.
[32] Iansiti, M. Levien, R. "Strategy as ecology." *Harvard Business Review*. March 2004, pp. 68-79.
[33] Begley, T., Boyd, D. "The need for a corporate global mind-set." *Sloan Management Review*. Winter 2003, pp 25-33.
[34] Kumra, G., Sinha, J. "The next hurdle for Indian IT" *McKinsey Quarterly*, 2003 Special Edition, No. 4.
[35] Kern, T., Willcocks, L., van Heck, E. "The winner's case in IT outsourcing." Winter 2002, pp 47-70.
[36] Useem, M., Harder, J. "Leading laterally in Company Outsourcing." *Sloan Management Review*. Winter 2000, pp 25-36.
[37] 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.
[38] Drucker, P. "They're not employees, they're people." *Harvard Business Review*. Feb 2002, pp 70-77.
[39] 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.

[40] Elenkov, D. "Can American management practices work in Russia?" *California Management Review*, Summer 1998, pp 133-157.
[41] Johnson, M. "Learning from toys." *California Management Review*. Spring 2001, pp 106-125.
[42] Barhtelemy, J. "The hidden costs of it outsourcing." *Sloan Management Review*. Spring 2001, 60-69
[43] Lei, D., Slocum, J. "Global strategy, competence building and strategic alliances." *California Management Review*, Fall 1992, pp 81-97.
[44] Tallman, S., Fladmoe-Lindquist, K. "Internationalization globalization and capability based strategy." *California Management Review*. Fall 2002, pp 116-136.
[45] Powell, W. "Learning from collaboration: Knowledge and Networks in the Biotechnology and Pharmaceutical Industries." *California Management Review*. Spring 1998, pp 228-240.
[46] Davenport, T., Klahr, P. "Managing customer support knowledge." Spring 1998, pp 195-208.
[47] Quinn, J. "Strategic outsourcing - leveraging knowledge capabilities." *Sloan Management Review*. Summer 1999.
[48] Porter, M. "Changing patterns of International Competition." *California Management Review*. Winter 1986, pp 9-40.
[49] Earl, M. "The risks of outsourcing IT." *Sloan Management Review*. Spring 1996, pp 26-32.
[50] 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.
[51] Magretta, J. "The Power of Virtual Integration: An Interview with Dell Computer's Michael Dell." *Harvard Business Review*. Mar-Apr 1998, pp 73-84.
[52] Seshasai, S. and A. Gupta. "A Knowledge Based Approach to Engineering Design." *AIAA Journal of Spacecrafts and Rockets*. Vol. 41, No. 1. January-February 2004.
[53] M. Bohanec and B. Zupan "A function-decomposition method for development of hierarchical multi-attribute decision models." *Decision Support Systems*, Vol. 36, No. 3. January 2004, pp 215-233.
[54] J. Wu, H. Doong, C. Lee, T. Hsia and T. Liang. "A methodology for designing form-based decision support systems." *Decision Support Systems*, Vol. 36, No. 3. January 2004, pp 313-335.
[55] De Neufville, R., Applied Systems Analysis: Engineering Planning and Technology Management, McGraw–Hill, New York, 1990.
[56] Srinivasan, V. "O Outsourcing and Offshoring" Presentation to MIT Special Seminar on International Management - Offshoring, Feb. 25, 2004
[57] Saini, Sanjay. "Offshoring Experiment in Radiology". Presentation to MIT Special Seminar on International Management - Offshoring, Feb. 25, 2004.
[58] Suh, Bob. "Perspectives on Global Delivery". Presentation to MIT Special Seminar on International Management - Offshoring, Feb. 25, 2004
[59] Malhotra, Raj. "Unique Business Model." Presentation to MIT Special Seminar on International Management, Apr. 10, 2004.
[60] Ellis, John. "Building an offshore contract engineering business." Presentation to MIT Special Seminar on International Management – Offshoring, Apr. 10, 2004.
[61] Shah, Swapnil. "Offshore Development." Presentation to Massachusetts Software Council, Apr. 13, 2004.

[62] Swadia, Sandeep. "Increasing Software Development Efficiencies." Presentation to Massachusetts Software Council, Apr. 13, 2004.
[63] Baxter, Stephan P."A Business Primer: How to Expand into China". Presentation to MIT Special Seminar on International Management - Offshoring, Apr. 10, 2004.
[64] Andre, Dave. "Case Study: Technology Offshoring at Upromise". Presentation to Massachusetts Software Council, Apr. 13, 2004.
[65] Sukharev, Alexis. "Auriga". Presentation to Massachusetts Software Council, Apr. 13, 2004.
[66] Terdiman, R., and Young, A. *Trends in application outsourcing for 2003 and 2004*, Gartner group, January 2003 (2003).
[67] Mukherji, S., and Ganguly, A. R. (2004). Sustaining the offshore outsourcing boom for software development: transitioning from low-cost service providers to strategic partners for information systems. Proceedings of the 9th International Symposium on Logistics (9th ISL): Logistics and Global Outsourcing. (To be published).
[68] Ferdows K "Making the Most of Foreign Factories", Harvard Business Review, March-April 1997.
[69] Interviews conducted by the author, in person, and via phone and electronic mail, with members of U.S. and Indian software development team, from April 9 to April 14, 2004.
[70] Hu and Grove, Encountering the Chinese: A Guide for Americans. Intercultural Press; 2nd ed. January 1999.
[71] Yap, M., "Follow the sun: Distributed extreme programming development," *Agile Conference, 2005 Proceedings*, pp. 218–224, 2005.