DOI: 10.1145/1378727.1389973

BY JULIA KOTLARSKY, ILAN OSHRI, KULDEEP KUMAR AND JOS VAN HILLEGERSBERG

Towards Agility In Design In Global Component–Based Development

The potential benefits of implementing Component-Based Development (CBD) methodologies in a globally distributed environment are many. Lessons from the aeronautics, automotive, electronics and computer hardware industries, in which Component-Based (CB) architectures have been successfully employed for setting up globally distributed design and production activities, have consistently shown that firms have managed to increase the rate of reused components and sub-assemblies, and to speed up the design and production process of new products.

CBD can indeed be an appealing proposition to globally distributed software development teams because of the possibilities offered to project teams to distribute work in such a way that each site takes responsibility for the development of a particular component. Indeed, it has been suggested that CBD will improve software development practices by allowing each site to assume ownership of particular components, resulting in reduced inter-site communication and coordination.[3, 11] However, such an approach, in our view, may create fewer opportunities for component reuse, as limited inter-site communications will impair knowledge sharing activities, thus diminishing possibilities to achieve agility in design.

In this article we explore how two globally distributed CB project teams coped with inter-site coordination and communication challenges while still improving agility in their designs. Against past studies expectations, these companies opted to utilize the expertise available to the dispersed teams regardless of their geographical location; and developed capabilities that improved agility in design. By considering the experiences and capabilities described in this paper, we hope that other companies may contemplate how agility in design can be achieved despite geographical distances and cultural diversity.

In the software industry, CBD is a relatively new trend. It emerged in the mid-90s with the introduction of software component technologies such as Enterprise JavaBeans, Microsoft COM and CORBA. *Component-Based (Software) Development* involves (i) the development of software components and (ii) the building of software systems through the integration of pre-existing software components (developed in-house or procured from the component market).[1] CBD was presented as a revolutionary approach in the software industry, promising dramatic improvements in software development, such as the endless possibilities to reuse and recombine software components, shorter time-to-market and reduced development cost.[4, 5, 10, 12] In this regard, CBD offers agility in de-

sign by basing software development on methodologies that support the recombination of reusable components,[2] being an approach that rapidly expands product variation and sustains the build-up of product families. Along with this positive outlook, some possible challenges that co-located software development teams have faced have been reported in the literature. For example, it has been argued that 'it often took longer to develop a reusable component than to develop a system for a one-off purpose'.[5] Other studies have reported difficulties associated with the management of CBD in co-located projects, such as a lack of stable standards, lack of reusable components, and problems related to the granularity and generality of components.[4,12]

The adoption of CBD by globally distributed software development teams has created expectations that some coordination breakdowns associated with the traditional, non-CB globally distributed software development would be mitigated.[3] For example, it was argued, each site could take ownership of a particular component to develop it independently, with a reduced degree of inter-site communication and coordination.[11] Indeed, this process would facilitate the globalisation of the software industry, as components could be developed remotely with minimum coordination across dispersed locations.[3]

While such anticipations relating to CBD in globally distributed teams is encouraging from the software development perspective, it is argued here that from agility in design and the possibilities to reuse existing components, such an approach in which each site takes ownership of the development of a particular component may actually result in fewer possibilities to reuse components and diminishing opportunities to build agility in design.[7] Research has indeed shown that knowledge embedded in a design can be contextual and specific to a particular business context, to the extent that the transfer of such solution between project teams often requires intensive communications between members of these teams.[9] Against past expectations that CBD will result in fewer communication and coordination activities, we claim that, in pursuing CBD principles, remote teams are more likely to intensify their communication and coordination activities should they wish to enjoy the benefits offered by CBD, which are mainly in the form of component reuse and product variation. Table 1 summarizes the expectations from globally distributed CBD and the new challenges for component reuse and agility in design.

Based on evidence collected at TCS and LeCroy, we have observed that in order to cope with the challenges outlined above, managers considered two key issues that affect the way agility in design can be realized in their globally distributed CBD environments. The two key questions were:

1. Which division of work approach (i.e. one site owning a particular component or joint development regardless of the geographical location of expertise) improves agility in design in globally distributed CBD teams
2. What capabilities were needed to support agility in design in their globally distributed CBD teams

Following a short introduction to the projects, we report on the approach taken in terms of division of work and the capabilities developed to support agility in design in globally distributed software development teams such as TCS and LeCroy.

TCS: Globally Distributed CBD

The project studied at TCS concerns the development and implementation of Quartz, an integrated financial platform aimed at providing solutions for financial institutions such as traditional and internet banks, brokerage/securities houses and asset managers. Quartz consists of a collection of architectural and business components that can be integrated with third party components to provide a solution according to the requirements of a specific customer. A typical Quartz implementation project includes the development of new customer-specific components, modification of existing components and integration of some pre-developed components with the customer's systems. We investigated the implementation of Quartz for Skandia bank, located in Switzerland. The development team were distributed across three geographical locations: two *offshore* teams in Gurgaon and Bombay (India), and an onsite team at the customer location in Zurich (Switzerland). Furthermore, vendors of third party components were located in various countries (more than 25 vendors in total).

LeCroy Globally Distributed CBD

The project studied at LeCroy, called Maui, involved a development of a software platform for new generations of oscilloscopes and oscilloscope-like instruments based on the Windows operating system.[6] The LeCroy software team was distributed across three sites: two teams located in Geneva (Switzerland) and New York (USA), and a main architect telecommuting from Maine (USA). One key challenge for the Maui project was to switch from embedded software to Microsoft COM technology.

Expertise-Based Or Location-Based Division Of Work

Despite the expectation that CBD would enable a division of work based on *geographical location*, in which each site takes responsibility for a particular component thus reducing the need for inter-site coordination required to complete a particular development,[11] evidence from TCS and LeCroy suggests that dividing the work based on *technical or functional ex-*

Table 1: Expectations from Global CBD and the emerging challenges for component reuse and agility in design

Expectations	Emerging challenges to achieve component reuse and agility in design
▸ Each site could take ownership of particular components and work on them independently without much need for inter-site coordination. ▸ Opportunity to reduce communication needs between sites is working on different components. ▸ Intensive knowledge processes withing remote site and fewer across sites.	▸ Remote sites could lose a "bird-view" of new developments, i.e. new developments in other sites are less visible. ▸ Differences in terms of levels of technical expertise developed within sites that could hamper the quality of knowledge transfer preocesses from team A to B when a particular component is reused. ▸ Each site could specialize in a different domain or market (e.g. site A specializes in the banking sector while site B specializes in the insurance sector). This could hinder the development of complementary solutions of neighboring sectors and implementation of agility in design that offers future integration of existing solutions. ▸ Difficulties to manage complex integration processes of reusable components because tacit and contextual component knowledge resides only within the develpoping site.

pertise has resulted in a high rate of component reuse and product variation. In addition, a division of work based on expertise enabled TCS and LeCroy to utilise the knowledge and expertise of their employees regardless of their geographical location, thus allowing these companies to access the pool of expertise available in offshore locations. At the same time, this approach required remote engineers and managers to intensify their communication and coordination activities.[2]

In dividing the work between the sites, certain ground rules were followed. For example, at TCS the first activities that required direct customer contact and access to the customer's site, such as user requirements and release management, were done onsite. Self-contained activities, such as coding and unit testing, were on the other hand, sent *offshore* to take advantage of the cost, quality, and availability of offshore personnel. As one interviewee explained, the onsite team was sending requirements offshore 'because the expertise and major source code are here [offshore, in Gurgaon], and mainly because of the expertise, it is quicker and easier to work here'.

Finally, activities that required involvement of the client and close interactions among the TCS developers were conducted in a mixed *onsite-offshore* mode. Overall, the majority of the activities required extensive communications, and onsite and offshore teams conducted them jointly through frequent communications using ICTs and visits to remote sites. The only activities that were done independently at the offshore location were coding and unit testing.

At LeCroy, software engineers specialised in different technical domains. One engineer explained: 'each of us know really well one part of the system, so we have kind of specificities, we know better one domain than another one'. Consequently, the division of work was also based on the technical domains of the engineers. This approach to division of work at LeCroy, in which engineers worked jointly on development, required a high utilization of the Integrated Development Environment and the employment of collaborative technologies. As the manager of Geneva team pointed out:

We use MSN messenger - every member of the software development group appear on the list. So for having a chat with someone, wherever they may be in the world in the given time, you just need to double-click on their name and start typing a line.

Capabilities Supporting Design Agility In Globally Distributed Component-Based Development

An expertise-based division of work approach posed new challenges to TCS and LeCroy. In particular, as these companies attempted to improve product agility in their globally distributed CBD teams, three areas had become critical in achieving success: inter-site coordination, knowledge management, and communication channels.

Inter-site coordination capabilities. Two areas of capabilities were developed by TCS and LeCroy that supported inter-site coordination activities. The first area, which is generic in nature, involved a high-speed, wide-band ICT infrastructure that ensured connectivity between remote sites. The ICT infrastructure supported rapid access to the network, shared resources (e.g. server and project repository, databases) and Web access to common resources. The second area of capabilities was developed mainly in-house to support joint development activities carried out across remote sites. These included the following capabilities:

▸ *Automated management of interdependencies between components and related files.* This capability allows for managing dependencies between components. Managing these interdependencies is not a problem as long as the number of components is small, because in such a case the dependencies can be modelled and understood visually. However, when the number of components becomes high, visual understanding is no longer an option. To manage interdependencies between a high number of components, LeCroy developed an in-house tool that supported rapid update of changes, four times a day, by automatically building components that changed after the last "build", thus utilizing time-zone differences. One manager from LeCroy explained:

We have a server that builds four times a day all the components, and produces 'release binaries'. So I don't have to build every component locally. If someone changes the hardcopy component and they put it back, it will be rebuilt on the server and then in the morning I can import that component and just use it.

This capability allowed developers from NY and Geneva access to the latest versions of the development files when they started their working day.

▸ *Bug tracking capability.* The complexity involved in developing components from several remote locations created new challenges in terms of tracking bugs. This is because each single bug being reported required its source to be tracked and traced (i.e. whether it had originated from one of the customers or from an internal development team), and required the location of all the components in which this particular code was reused. For bug tracking, TCS used a combination of tools: MS-Access for issues registration and resolution, and PVCS Tracker for defect logging, tracking and analysis.

▸ *Standardization and centralization of the tools and methods* across locations was perceived as imperative by TCS and LeCroy for the effectiveness of component reuse, mainly because software development activities such as building and testing a code can be carried out on the central server. Some of the technologies implemented to facilitate this capability are: Web access, replicated databases, and a single (integrated) development environment. To facilitate the standardization of processes and learning among newcomers, TCS and LeCroy created a Guide that explains how to use tools and methods.

Knowledge Management

Ensuring a high degree of component reuse and agility requires knowledge management capabilities. One key challenge that TCS faced, for example, was that several Quartz implementation projects for different clients were running in parallel and the people involved in different Quartz implementations did not have a direct exposure to the work that other project teams were engaged in. Therefore, several knowledge management mechanisms were introduced by TCS that facilitated the transfer and reuse of components across product teams. For example, TCS created a new role in the form of a program manager, who coordinated all Quartz implementations and was aware of new components being developed for a specific customer, and who facilitated reuse of components across different Quartz implementations at different geographical locations. This way TCS exploited customer-specific compo-

nents by adding them to the Quartz package, so that with each new Quartz implementation TCS increased the variety of components / functionalities that this product offered to potential clients. For example, after implementing Quartz at Royal Skandia UK, an insurance company, where Quartz was implemented as an investment engine, insurance products were added to the next version of Quartz.

LeCroy, on the other hand, developed a 'component toolbox', a *repository of components* that implemented functionalities common to the oscilloscopes and oscilloscope-like instruments they develop. These components included hundreds of mathematical functions, one component per functionality; Graphical User Interface components that provide the user interface; core components that allow the systems to work together and provide the basic instrument capabilities; and acquisition board driver components responsible for controlling the acquisition hardware of an oscilloscope. For example, the "Touch Screen Calibration" icon on LeCroy's WaveSurfer oscilloscope is managed by a particular software component. Touching this icon on the screen will activate a calibration procedure.

Based on this knowledge management system, a specific oscilloscope could be built by selecting and integrating these components with oscilloscope-specific acquisition and application systems. Furthermore, LeCroy introduced an integrated development environment on the central server, accessible for all members of the dispersed team.

Communication Channels

While coordination mechanisms were put in place to allow remote counterparts to access data imperative for the joint development process, communication channels were implemented to ensure that members of a dispersed team would also be able to discuss and exchange knowledge about design considerations and any solutions implemented. Indeed, developing communication channels and capabilities that ensure the flow of information with minimum breakdowns and misunderstandings between remote sites was also a critical factor in achieving design agility in globally distributed teams. To cope with this challenge, both TCS and LeCroy encouraged *frequent communications* between remote members and introduced design rules that made communications more effective. For example, these companies encouraged systematic and frequent communications in the form of regular teleconferences between software managers in dispersed locations, and transatlantic videoconferences with the entire team every one or two months. This helped to streamline information flows between dispersed teams. In addition, attention was given to *improving the style and content of communications*, which helped to reduce misunderstandings and confusion induced by different cultural backgrounds. This was particularly important for the LeCroy global team, where people from different cultures collaborated over distance.

Both companies encouraged *working flexibility*, in terms of working conditions, e.g. working from home, and working hours, which helped increase the overlap of working hours between dispersed locations so that teams could collaborate in real time.

As demonstrated above, the development of a successful agility in design in globally distributed teams requires the application of methodologies and tools that support intensive inter-site coordination and communication along with knowledge management strategies that ensure the reuse of that design. Table 2 summarizes the results of this study, offering managers practices that allow them to achieve agility in globally distributed CBD.

Achieving Agility

Both companies successfully and rapidly developed product families from their core platforms. LeCroy's Maui CB architecture (platform) served as a basis for a number of future products, helped to reduce considerably time-to-market (an increased number of products offered to the market per year), and made possible easy integration of LeCroy products with additional functionalities developed by third parties. After the first (WaveMaster) product based on the Maui architecture was launched in January 2002, a large number of WaveMaster models and Disk Drive Analyzers, which are based on the Maui platform, were released, all a result of a rapid recombination and reuse of existing components. In January 2003, LeCroy launched the WavePro Oscilloscope series, which is also based on the Maui platform.

TCS Quartz CB architecture also supported the reuse of components across different implementations by adding customer-specific components to the Quartz package. Since the first Quartz implementation project started in 1998, Quartz has been implemented in over 40 clients, and the platform has grown to include 3 different product families: Quartz Securities, Quartz Payments and Quartz Financial. Furthermore, in recent years, Quartz was ranked among the top 25 best-selling banking systems by the International Banking Systems Journal.

Against the expectations of past studies, this study illustrates how globally distributed CBD projects may make the most of component reuse opportunities by pursuing an approach in which remote sites jointly develop components and by utilizing expertise regardless of its geographical location. This was achieved by developing capabilities in three particular areas: inter-site coordination, communication channels and knowledge management.

We further claim, based on evidence presented in Table 2, that an approach in which components are jointly developed by dispersed team members regardless of their geographical location, enables the connecting of 'islands of knowledge' that are otherwise isolated by geographical, time-zone and cultural differences. As shown in Table 2, *knowledge management* practices directly contributed to achieving agility in design through repositories and program managers that together ensured the capturing and dissemination of component knowledge for reuse in new products and services. Through *inter-site coordination* mechanisms such as the automated management of interdependencies between components and standardized tools, a uniform development environment across sites was created and integratability between components was ensured. Indeed, remote counterparts' ability to easily access and reuse components was enhanced mainly because of their past participation in development and debugging activities across sites, which expanded their familiarity with the component or the person who developed it. The various *communication channels* (e.g. videoconferencing, chats) put in place supported agility in design by creating imperative conditions for collaboration and knowledge sharing despite language barriers, geographical distance

and cultural differences. Indeed, the process of component reuse could only benefit from communication patterns that rely on a shared understanding of technical and business contexts developed within a dispersed team.

Nonetheless, we acknowledge that our findings are based on only two case studies and therefore, by definition, meet to only a limited extent the criteria of transferability (i.e. the extent to which the findings can be replicated across cases). However, we hope that based on the experiences and practices reported above, other companies could improve agility in design when implementing CB principles in their globally distributed teams.

This study also demonstrates that there is interplay between product and process agility. Firms that attempt to improve product variation through CBD methodologies also need to consider introducing higher flexibility in their coordination and communication processes. Similarly, firms that follow agility in design methodologies in their software development processes may also consider building greater flexibility in their products by implementing CBD methodologies. Ideally, agility will be achieved in both software development processes and products. C

Table 2: Achieving agility in globally distributed software development through CBD

Capabilities	Characteristics	Benefits for enabling agility in design
Inter-site coordination capabilities		
Automated management of interdependencies between components and related files	Automatically building components that have changed since the last "build", several times a day, on the central server	Avoiding "re-inventing the wheel" by providing remote counterparts access to work done at remote sites and making them aware of reusable components that are being developed elsewhere
Bug tracking across products and projects	Setting up a bug tracking tool on the central server	By tracking all components and products affected by a bug, the pool of components is always kept updated with latest fixes, thus avoiding having different versions of the same component within the component repository
Standardization of tools and methods across locations	Using similar tools and methods across dispersed locations, creating a Guide that explains how to use methods and tools	Ensuring compatibility and "integratability" of files, components and applications developed at remote locations to enable reuse across locations. Creating uniform development environment across sites to facilitate reuse
Centralization of tools	Centralizing tools used by dispersed team members on a central server, Web-access, Integrated Development Environment	Creating a common development infrastructure across remote sites that supports the exchange and reuse of components.
Knowledge management capabilities		
Component repository accessible from all dispersed locations	Creating repository of reusable components that implement functions common to majority of products	Enabling knowledge reuse across products. Reducing time-to-market through the reuse of existing components Increasing product variety
Program manager	Appointing program manager to have overview of all components being developed for different clients	Supporting the sharing of implicit component knowledge across projects Increasing product variety through the sharing of business knowledge
Communication channels and capabilities		
Frequent communications	Organising systematic and frequent communications between managers and developers of dispersed teams using on-line chat, email, teleconferencing and videoconferencing	Creating opportunities to bridge knowledge gaps that may hamper knowledge transfer processes Streamlining information flows (who knows what)
Improving the style and content of communications	Adjusting style and content of communication (e.g. wording and selection of media) to personal and cultural characteristics of remote counterparts	Enabling knowledge sharing processes between remote counterparts Reducing possibility of misunderstandings and conflicts between remote counterparts
Working flexibility	Supporting flexible working conditions, e.g. equipment to enable working from home and flexible working hours	Enable remote counterparts to collaborate in real time by increasing overlap in working hours

References

1. Bass, L., C. Buhman, S. Comella-Dorda, F. Long, J. Robert, R. Seacord, K. Wallnau, *Volume I: Market Assessment of Component-Based Software Engineering*. Carnegie Mellon Software Engineering Institute (SEI), 2000.
2. Boehm, B.W., R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*, Boston: Addison-Wesley, 2004, 216-229.
3. Carmel, E., *Global Software Teams: Collaborating Across Borders and Time Zones*, 1st ed, Upper Saddle River, NJ: Prentice-Hall P T R, 1999.
4. Crnkovic, I., M. Larsson, *Challenges of Component-Based Development*, The Journal of Systems and Software 61(3), 2002, pp. 201-212.
5. Huang, J.C., S. Newell, R.D. Galliers, S.-L. Pan, *Dangerous Liaisons? Component-Based Development and Organizational Subcultures*, IEEE Transactions on Engineering Management 50(1), 2003, pp. 89-99.
6. Kotlarsky, J., *Reengineering at Lecroy Corporation: The Move to Component-Based Systems*, Journal of Information Technology 22(4), 2007, pp. 465-478.
7. Kotlarsky, J., I. Oshri, J. van Hillegersberg, K. Kumar, *Globally Distributed Component-Based Software Development: An Exploratory Study of Knowledge Management and Work Division*, Journal of Information Technology 22(2), 2007, pp. 161-173.
8. Kunda, D., L. Brooks, *Assessing organisational obstacles to component-based development: a case study approach*, Information and Software Technology 42, 2000, pp. 715-725.
9. Oshri, I., S. Newell, *Component Sharing in Complex Products and Systems: Challenges, Solutions and Practical Implications*, IEEE Transactions on Engineering Management 52(4), 2005, pp. 509-521.
10. Ravichandran, T., M.A. Rothenberger, *Software Reuse Strategies and Component Markets*, Communications of the ACM 46(8), 2003, pp. 109-114.
11. Repenning, A., A. Ioannidou, M. Payton, W. Ye, J. Roschelle, *Using Components for Rapid Distributed Software Development*, IEEE Software 18(2), 2001, pp. 38-45.
12. Vitharana, P., *Risks and Challenges of Component-Based Software Development*, Communications of the ACM 46(8), 2003, pp. 67-72.

Julia Kotlarsky (jkotlarsky@wbs.ac.uk) is Associate Professor of Information Systems in the Information Systems and Management group at Warwick Business School (UK).

Ilan Oshri (ioshri@rsm.nl) is Associate Professor of Strategy and Technology Management at the Department of Strategy and Business Environment, Rotterdam School of Management Erasmus (The Netherlands).

Kuldeep Kumar (kumark@fiu.edu) is a Visiting Professor of IS at The City University of Hong Kong (China). He is also the Professor of IS at Florida International University, Miami (USA), and Professor of IS Research at RSM Erasmus (The Netherlands).

Jos van Hillegersberg (j.vanhillegersberg@utwente.nl) is a Professor of Design and Implementation of Information Systems and head of the Department of Information Systems and Change Management at the University of Twente (The Netherlands).

© 2008 ACM 0001-0782/08/0900 $5.00