Collaboration, Knowledge and Interoperability — Implications for Software Engineering

Norbert Jastroch
MET Communications GmbH
Eschbacher Weg 10
61352 Bad Homburg, Germany
norbert.jastroch@metcommunications.de

Vassilka Kirova
Alcatel-Lucent
600-700 Mountain Ave.
Murray Hill NJ 07974, USA
vassilka.kirova@alcatel-lucent.com

Thomas J. Marlowe
Department of Mathematics and
Computer Science
Seton Hall University
South Orange NJ 07079, USA
thomas.marlowe@shu.edu

***Abstract*—As software development becomes more collaborative, all aspects of software engineering and their management need to accommodate and support collaboration. In this paper we present an updated survey of key concerns, known challenges, and potential alternative solutions, addressing a number of new issues and opportunities.**

***Keywords*- Software Engineering, Collaborative Platform, Flexibility, Interoperability, Business Policies.**

I. Introduction

Inter-organizational cooperation in software product and tool development has proliferated throughout the industry and has become the "de-facto" business model [15, 30]. Continuing trends for outsourcing and offshoring, subcontracting, and academic-industrial collaboration, plus increasing sophistication and specialization of business software (and for that matter, most of knowledge-intensive software), make an ever stronger case for close inter-institutional collaboration throughout the development process. While legal constraints and environmental support, along with a number of management issues, appear to be the most substantial challenges, we contend that all aspects of software development need to be re-examined and revised to effectively address end-to-end collaboration. In this paper, we consider collaborative development of large, complex systems, with large or complex components, which may however be feature- or service-based, rather than resulting from a functional decomposition, and in contrast to projects that can be handled as supply-chain or other simpler structures [34,42].

A key challenge in multi-organizational collaboration efforts is that institutional and team responsibilities need to be assigned early in the project inception phase, which requires the scoping, decomposition of the product and the definition of the overall project structure to be established well before the application and often the technology to be developed is even marginally understood, but after the parties have agreed to enter into a collaborative venture.

That, especially in large, complex or innovative projects, in turn requires flexibility in defining the scope and allocation of responsibilities in interactions between teams and partnering institutions. In development, it further requires flexibility in defining component, subsystem and feature boundaries, understanding that the best high-level decomposition may not be the optimal, but also be driven by management objectives, technical expertise, resources, and other considerations in the individual organizations, and perhaps as importantly, by the relative stability and simplicity of the interfaces between partner components. Any tradeoffs evidently should not devalue the collaboration below any established cost/benefits threshold or jeopardize the system objectives. This is a critical consideration in assembling the collaborative venture at the start.

A notable current trend in both the technical and management aspects of software development is flexibility. In the requirements and development segments, this often involves some form of agile development. Agile frameworks accommodate changing requirements through continuing customer involvement, self-organizing closely collaborating cross-functional teams, short iterations, application of technical practices such as test-driven and acceptance test-driven development, test automation and continues integration, as well as through avoiding unnecessary specification and upfront extensive analysis. However, agile development may not work well as the sole process model even within a single organization [2, 6]. Agile methods bring the much needed flexibility but given the constraints of multi-organizational collaboration, we expect that instead a typical collaborative development effort will support local agility within partner components, paired with a more tightly controlled flexibility at organizational and corresponding component and subsystem boundaries and interfaces.

In the body of the paper, we very briefly consider a number of policy, artifact, and process issues that need to be resolved to foster and optimize both the eventual products of the collaborative venture and the health of the collaboration itself. Addressing these issues complements rather than competes with selection and establishment of a good collaborative platform, development, production and enterprise-level social and tool environment. Tools address the question of how collaborative work is to be supported,

whereas our concern here is focused on what needs to be done to allow the process, product and partnership to succeed, and to provide a new scientific, technological, and business practices asset base to allow software enterprises and in particular small-to-medium software engineering and computer-based application firms to collaborate and to establish themselves as innovation drivers and significant players in the area of future Internet enterprise systems. (This paper revisits and extends an earlier survey, presented at ICSSEA 2010 [14], in part to incorporate a great deal of recent work on interoperability and the use of the cloud for collaboration, and in part as a result of additional work by us and by others in the intervening two-and-a-half years.)

II. Early Business Policy Decisions and Interoperability

In a collaborative software development project, several key decisions must be made prior to anything but the earliest vision of a system/product and its business case, either during project initiation by the individual partners, or very early in the inception phase. The first, of course, is determining whether to collaborate, how to collaborate, and with whom [34]. Assuming a fully collaborative project, decisions must be made in several dimensions, before or during project inception: responsibilities in the collaboration, resource acquisition and management, establishing or reinforcing trust and familiarity, process and platform consistency, and protection of security, intellectual property, and general interests of the partners and other stakeholders [35, 37].

Responsibility and resource issues include (1) a policy on sharing of resources, including key personnel, and on cross-organizational discipline-specific collaboration [7], and (2) responsibility for the cost of new shared tools, personnel, and resources. Business project support includes (1) establishing risk management plans and management contingency policies for the collaboration, and (2) establishing a venue or process for mediation and arbitration, as well as (3) creating inter-organizational plans for cross-sensitizing, communication and training, and (4) establishing a procedure for interaction with the customer and other stakeholders—who should not be confused and overwhelmed by multiple contacts with different organizations.

Process and platform technical support includes (1) standardization of platforms, tools, and processes—or at a minimum alignment and support of standard and consistent views, (2) creating a shared software configuration management environment, and guidelines for sharing of artifacts, and (3) determining a level of agility and a level of formality—locally and at interfaces [9, 16, 28, 39]. Finally, collaboration must be supported by seeking and maintaining steady buy-in from management, technical staff, IT and legal departments, while respecting the interests of the individual partners and stakeholders.

In recent years, the concept of interoperability [5, 19, 31] has grown from a focus on technical compatibility to include cultural, knowledge and risk, business and strategy, and ecosystems interoperability, and to address communication in a very broad sense—culture, trust, social networks and other social exchanges—as well as security and use of the cloud. The cloud is important for collaboration, providing a (potentially secure) location for the location of collaborative resources, artifacts and work products, including collaborative configuration management and a neutral and (ideally) trusted site to measure activity for the purposes of credit and use assessment, at least of explicit knowledge.

Finally, collaboration introduces a much wider sense of and scope for exceptions [32], not only in the discovery and handling of run-time exceptions and exceptional flows during software development and deployment, but also for business and technical practices and processes, knowledge and risk management, and other phases and aspects of a collaborative project. Beyond those that arise from product requirements, one will need to handle exceptions arising from incomplete or inexact specification of partner rights and responsibilities, misunderstandings or failures in fulfillment of those rights and responsibilities, and failures to establish true interoperability in some—not necessarily technical—aspect.

III. Requirements and Knowledge Management

Incomplete, imprecise or changing requirements interact with the initial architectural decomposition, affected by missing knowledge, dynamic information, or the need for knowledge integration and synthesis. Hence, means for iterative discovery, analysis, knowledge building, and delivery become essential. First, support for dynamism in defining features and components is needed [41]. Collaborative projects, because of their highly-distributed work processes, must employ mechanisms for continuous requirements clarification between separated development teams, and support knowledge flow between partners and with collaborative artifacts and interfaces [16, 27, 28]. As a substitute to face-to-face communication, which is hard to establish and maintain in such projects, enhanced requirements tracing should be used to afford an efficient means to clarify requirements and address the related risks of misinterpretation [11, 13].

Second, approaches must be developed for collaborative knowledge discovery and integration, and collaboration in detecting and proactively managing unknown risk (compare [33, 34, 37]), while addressing risks to security, privacy and intellectual property, either inherent in the sharing and integration of knowledge, or from this coordination itself. Third, complex systems working within dynamically changing environments are influenced by inbound influences and outbound impact—beyond their functional interfaces—that cannot be fully anticipated, but to which they need to adapt. These inbound and outbound factors may introduce risks or opportunities, and may be well, or partially, or even not at all understood in advance. As result, support for adaptability becomes a major operational requirement and consideration.

A clear need is to link flexible approaches, including agility, from software engineering frameworks with the collaborative trust paradigm from the organizational sciences, knowledge exploration, augmentation and exploitation concepts from knowledge management, and ideas from interoperability, cloud computing, the Internet of services, and software-as-a-service.

Finally, the issue of ownership of integrated, collaborative and emergent knowledge, rights to the use of intellectual property generated by the collaboration, and mutual obligations in the knowledge framework, will inherently involve technical, business and legal considerations [29].

IV. Architecture, Analysis and Design

The large-scale organization of the product—its architecture—is partially determined by (or along with) the high-level decomposition and definition of interfaces between components developed by different parties—compare [12]. Although the parties could use different architectures within their own components, interoperability is widely recognized as a necessity for successful collaboration. Uniformity (or at least consistency) of architecture, of process and of development platform increases

communication bandwidth through shared vocabulary and mental maps and reduces miscommunication; supports better software configuration management and flexible interfaces, facilitates knowledge discovery and integration; and enables evolution, scalability and other non-functional properties. Most collaborative ventures will benefit from a scalable architecture with some measure of agility within components and flexibility across their boundaries. Uniformity of platform and practice may be facilitated through the cloud and interoperability standards such as [5], but these must be supplemented with collaborative artifacts and structures, and through examination of a number of the issues discussed here.

Agile development practices, if used, will have impact as well. [6, 8] suggest approaches for extending agility to widely distributed and collaborative ventures. Assuming an incremental iterative (if not fully agile) collaboration requires changes both in artifacts and in the activities. The most important from an architecture and design point of view is that there will be two distinct if sometimes overlapping workflows: the first, determining relatively stable high-level components, boundaries and interfaces, and a second, determining the internal organization of those components, both of which can run in iterations gradually addressing additional aspects, features and requirements or adding more details.

If feature-based partitioning of the work is being used (preferred by some agile teams and organizations) instead of component-based one, and features cross components' boundaries, then a common development environment, for example, cloud-based, becomes essential. In such cases component and interface guardians may need to be selected to assure coherent and consistent designs and implementation of individual components. If cross-cutting features are to be implemented with aspects [21], implementation needs either to be localized into individual components (and possibly interfaces), or raises another dimension of coordination and interoperability.

Inter-component interfaces must be flexible enough to support at a minimum handling of newly discovered exceptions and alternate flows, and to permit exchange of metadata and system information [17, 26, 28] for cross-component optimization and refactoring [8], without initially requiring communication of large amounts of such information, only a small fraction of which may actually be useful. Such interfaces can be extended to process and business objects, to support other activities and artifacts such as cross-component traceability and policy localization/specialization. On the other hand, the partition into components cannot be considered "sufficient to proceed" without at least some analysis of component responsibilities. These cannot be determined without considering both structural factors such as internal cohesion and cross-component coupling, in order to limit the cross-organizational footprint of change, and policy and resource factors such as division of labor and availability of expertise.

Achieving a consistent architecture in a collaborative project, however, is not a trivial issue even when standard tools, e.g., UML and SysML, are being used, and when knowledge and risk management are fully interoperable. The impact of defect models has to be investigated [22]. Furthermore, implications of model defects for implementation need to be understood.

V. Testing

Testing is inherently a collaborative activity [43]. It depends on close collaboration between teams and team members and often involves interactions with customers. Thus policy and governance rules similar to the ones related to requirements apply and need to be addressed accordingly.

Testing is ongoing, multilayered (unit, integration, acceptance) and should be automated. The organization of testing activities highly depends on the project partitioning model, the process models (development frameworks) used by the individual partners, the architecture of the product, the organization of the collaborative partnership, and the allocation of responsibilities. Unit level testing is essential and remains a local responsibility, while the responsibilities for integration and acceptance testing have to be agreed on as part of the collaborative partnership. Integration testing across organizational boundaries may require additional planning, testing effort, coordination, and sharing of testing strategies and resources. Not only does integration testing have to work across organizational boundaries, but in addition responsibilities for debugging and fixes must be assigned across those boundaries. Further, if more sophisticated and stateful interfaces or interaction patterns, such as those in [17, 27] are used, then interfaces themselves must be subject to unit testing. Also regression testing for changes that affect interfaces and possibly cross component boundaries will only be as effective as cross-component dependence analysis and traceability allow [17]. Acceptance testing reintroduces the issue of customer and user interaction and of indirect contact with the customer. Finally, to the extent that knowledge management, interoperability and/or collaboration require new or modified tools, those tools themselves will need to be tested—ideally, both alone and integrated with product components.

Any selected testing model will place specific requirements on the development, build (including continuous integration), revision control and change management environments. For example, continuous or early system and stress testing will require continuous integration or early integrated builds of the entire product. The effectiveness of the testing will depend not only on test automation, but also on good global revision control, change management, and traceability tools.

The testing practices have to be well-aligned with the requirements and feature definition and scoping methods and will depend on the development approach. And while full deployment of agile frameworks such as SCRUM [23, 24] across all partners of a multi-organizational collaboration project may not be feasible (due to organizational management, legal and other constraints), implementing agile test-related engineering practices is arguably a good fit. For instance, ATDD (Acceptance Test Driven Development) [24, 38] provides for good ongoing collaboration within the teams as well as with the customers.

VI. Metrics and Evaluation

There are four major issues for collaboration-aware metrics: the definition and selection of metrics, the gathering of data (measurement) for the metrics, and analysis and interpretation of the resulting measures, and finally the identification (and partner acceptance) of the need for additional focus and resources, and of required steps for remediation or improvement. Just as testing consists of unit testing, integration testing, and system/project testing, metrics for a collaborative project will include component metrics, organizational metrics (since an organization may be developing more than one component), and global metrics, for product, project, and process alike.

Evaluation for and within collaborative systems requires metrics that (1) accommodate collaboration, (2) measure readiness for collaboration and the effectiveness of the collaboration, and

(3) identify business policy/process and software process obstacles, and driving forces and enablers for collaboration [3, 18, 36]. Metrics for collaboration are needed for pre-collaboration, mid-collaboration, and post-collaboration, and must measure not only technical success, but also business and other aspects. For accommodation, existing metrics (whether used locally or for the entire project) must be modified or re-interpreted, and new metrics may be needed [20]. For example, effort and resource estimates must account for the impact of collaboration. On the other hand, cost estimates and measures can be used to evaluate the effectiveness and overhead of collaboration, and also to guide project restructuring and system refactoring. Still other existing metrics may no longer be useful in a collaborative setting, or may need to be retired or replaced.

Ideally, one would like to be able to estimate the cost of developing a project in isolation, and compare with the costs of the collaborative venture. Even an approximate heuristic may be difficult to establish, however, since additional resources and training, as well as opportunity loss if single-organization development took longer or used substantially greater resources, impose additional costs, and establishment of relationships with other organizations may create ongoing benefits, while the cost of loss of knowledge and intellectual property, especially implicit and tacit knowledge, may be hard to quantify, or to measure against the gain in knowledge from partners and the collaboration itself.

Readiness metrics must measure not only technical readiness, but management support for collaboration. Finally, while data collection is not a serious problem for local metrics, comparison of measures or computation of global metrics necessitates development of guidelines and practices for uniform collection. Responsibility for collection of the relevant information must be assigned (to an organization or a tool), and consideration must be given as to the proper encoding and weighting of the resulting measures.

VII. Maintenance and Evolution

Post-release responsibilities for maintaining a large, complex and strongly interacting system raise new issues for collaborative development. First, the lifetime of the system or product line may be longer than the lifetime of the collaboration. Also, a single change may require small changes in a large number of components. Finally, there is a timing issue: other than for major faults or security threats, changes can often be deferred. Key questions are (1) Who is responsible for determining that a change is needed, and when the change should be made? (2) How will the nature and locations of the needed changes be determined? and (3) Who will be involved in implementing and in testing the changed system? These questions tie into both the need for global traceability/dependence analysis [17] and arbitration [33], and especially questions of ownership and access to integrated, collaborative and emergent knowledge and intellectual property.

Perfective maintenance, as well as reuse, often involves refactoring. Local refactorings, assuming flexible interface structures (e.g., Façade and Adapter design patterns), is typically a local matter. But refactoring across interfaces or changing component responsibilities will again involve negotiation and determination of responsibilities, and significant intellectual property issues if one or more of the partners is no longer involved in the collaboration—and even more so if the partner is no longer doing business but retains IP interests in the product.

VIII. IT and Communication Support

Formal and informal communication links between partners form one of the four key factors in collaborative success, after management and technical competence, collaboration-aware risk management, and a collaboration-friendly management and technical environment. The success of any collaborative venture depends on availability of multiple forms of communication and meeting support, and technical and managerial support by IT departments and staff. The development environment should provide for sharing of code, test structure, design artifacts, and so on, preferably in shared tools or views. The Cloud and virtual environments provide additional opportunities and resources, with some associated risks.

IT is also responsible for implementing security, access control, intellectual property and privacy protections, knowledge management, etc. Typically a layered or hierarchical permission structure will be required, allowing sharing of some information (or summaries) with collaborators but not outside world, as defined by rules and policies in the collaboration agreement. As mentioned above, the cloud may provide an opportunity to avoid some risks, by isolating collaborative knowledge and structures away from partner resources. This issue needs a great deal of future attention.

IX. Process Compliance and Optimization

Both product and process in software development are frequently held to standards, whether internal, industry-based, or required by the customer or by regulation. In the first three cases, both product and process compliance can follow a structure much like that we propose for risk management [33]: Internal compliance checking by each party, mutual checking at interfaces, and a mechanism for arbitration and negotiation. In some situations, review by a certified third party may also be required. Additional issues include: obstacles arising from internal standards, process or artifact inconsistency, and compliance scope across national boundaries. The cloud and interoperability standards are likely to introduce further compliance issues as their use becomes more standard and more regulated. Finally, technical process and business policy optimization resembles perfective maintenance—very important and usually beneficial, but neither urgent nor worry-free, and therefore requires a similar process.

X. Business Process Issues and Management Contingency Processes

Business management in a collaboration handles the standard tasks for management of software projects—personnel, budget and schedule tracking, and so on. Beyond these, in the literature and in our previous work [33, 35, 37], the most significant policy issues for collaboration are identified as (1) creating and maintaining trust, (2) handling differences in language and culture, both organizational and social, and (3) maintaining corporate support. In addition, management processes must assure the continuing quality of risk management and communication, and deal with problems and changes in the set of partners and with non-fulfillment of partner responsibilities. Corporate support for complex collaboration may need to overcome resistance from managers, lawyers and other professionals more used to the simpler demands of supply-chain collaboration. On the other hand, the growing use of the cloud may reduce IT and management resistance to some forms of sharing and to communication across institutional firewalls.

XI. Security, Privacy and Intellectual Property

Collaborative processes need knowledge from diverse sources, some of which raise specific security, privacy, or intellectual property concerns: product information, process information, platform and tool information, and recent corporate decisions and history. Use of such information, and more general concerns about such issues tend to harden management, IT, and legal expert resistance, and to inhibit collaboration. These issues must be addressed along both corporate (social/economic/legal) and technical dimensions.

On the corporate side, consider cost-benefit analysis for various levels of information sharing, with restrictions on external use of information gained. Sharing promotes trust and cooperation, with the risk of high rewards for low contributions. This suggests a differential approach, in which information is layered, and inner layers revealed only as a partner contributes. However, some knowledge must be shared *a priori*, since it will be required to initiate the collaborative process or product inception. In addition, changes in the set of partners can pose difficulties for this approach.

Technically, there are at least two aspects in alleviating these problems: first, selection or development of filters, abstractions, or views of information so that useful but safe summaries are available to collaborators and customers; and second, selection of publicly available or sharable tools and methods through which information can be imported and exported. These of course must complement use, undertaken and certified by all collaborators and other stakeholders, of standard secure mechanisms for data storage and information transmission, to address third-party threats. This is especially important for the development and communication platforms. Successful collaboration must rely on a shared and uniform view of important artifacts, and on rapid, reliable, and secure broad-spectrum communication, both formal and informal.

As mentioned above, the cloud and standards for interoperability are likely to help overcome resistance to controlled sharing of information, but do not, in our opinion, do much to address the difficulties arising from the need to share complex, structured information, the difficulty in handling integrated, collaborative, or emergent knowledge, or the problems arising from management of intellectual property. In addition, the cloud and artifacts required for interoperability do not resolve, and may well exacerbate, security and privacy concerns.

XII. Related Work and Conclusions

Most of the existing literature on collaboration considers intra-organizational collaboration, or focuses on selected aspects of inter-organizational interaction. This paper in contrast specifically addresses inter-organizational collaboration from a broad, multi-faceted and systemic point of view. Incorporating agile practices in intra- or inter-organizational collaboration is discussed in [5, 8, 10, 25, 26]. Herbsleb [12] and Whitehead [43] address the problems of collaboration, but largely within a single organization, and primarily limited to tool support and software configuration management, including expansion of the set of desirable artifacts; changes to management policy and perceptions, and in the software development process, are also discussed. Erickson [4] and Schadewitz [40] provide patterns for component interfaces and interactions. But, other than in our previous work, there seems to be little explicit focus on, for example, collaboration-aware metrics, changes in testing, knowledge management for collaborative software development, or risk management *per se*.

There are two other major sources of work that should be considered. First, there is a great deal of recent work on interoperability [5, 19, 31] and, second, on the use of the cloud for software development and for information sharing, including for collaborative development [1, 42].

In conclusion, we have addressed the implications of inter-organizational collaboration for a number of development aspects —including both core software engineering and umbrella activities. Inter-organizational development and collaboration for large, complex or innovative software products calls for new approaches to accommodate both organizational differences and flexibility in development. As we discuss, all aspects of software development—technical, process, and management—are affected. This paper outlines pressing issues and presents initial or partial solutions in several areas.

References

[1] A. Almutairi, M. Sarfraz, S. Basalamah, W. Aref, A. Ghafoor: A Distributed Access Control Architecture for Cloud Computing, IEEE Software, Volume 29, Number 2 pp. 36-44, March/April 2012.

[2] B. Boehm and R. Turner, Balancing agility and discipline: A guide for the perplexed. Addison Wesley Professional, 2003.

[3] C. Cook and N. Churcher: Modelling and measuring collaborative software engineering. 28th Australasian Conference on Computer Science, Newcastle, Australia, pp. 267-276, January 2005.

[4] T. Erickson, Interactions Design Patterns Page, http://www.visi.com/~snowfall/InteractionPatterns.html, 2010.

[5] European Commission: Enterprise Interoperability Science Base. http://cordis.europa.eu/fp7/ict/enet/fines-eisb_en.html, last accessed November 10, 2011.

[6] A. Filev, "Adopting and Benefiting from Agile Processes in Offshore Software Development," Microsoft Architect Journal, 2010, http://msdn.microsoft.com/en-us/library/bb245671.aspx

[7] D. Flynn, E. Brown, R. Krieg: A Method for Knowledge Management and Communication within and across Multidisciplinary Teams, KGCM 2008, June 2008.

[8] M. Fowler, Refactoring: Improving the Design of Existing Programs, Addison-Wesley, 1999.

[9] M. Fowler, Using an Agile Software Process with Offshore Development, 2010, http://martinfowler.com/articles/agileOffshore.html

[10] K. B. Hass, The Blending of Traditional and Agile Project Management, PM World Today, IX (V), May 2007, http://www.pmforum.org/library/tips/2007/PDFs/Hass-5-07.pdf

[11] M. Heindl, St. Biffl: Risk Management with Enhanced Tracing of Requirements Rationale in Highly Distributed Projects. Proceedings of the 28th International Conference on Software Engineering, 20-28 May, 2006, Shanghai, China.

[12] J. D. Herbsleb, Global Software Engineering: The Future of Socio-technical Coordination, 2007 Future of Software Engineering (FOSE'07), Minneapolis, Minnesota, May 23-25, 2007.

[13] N. Jastroch, T. J. Marlowe, Knowledge Transfer in Collaborative Knowledge Management: A Semiotic View, Journal of Systemics, Cybernetics and Informatics, Vol. 8 (6), pp. 6-11, 2010.

[14] N. Jastroch, V. Kirova, C. S. Ku, T. J. Marlowe, M. Mohtashami, Software Engineering Must Be Collaboration-Aware, (Position Paper), Proc. of the 23rd International Conference on Software and Systems Engineering and their Applications [ICSSEA], Paris, France, December 2010.

[15] N. Jastroch, V. Kirova, C. Ku, T.J. Marlowe, M. Mohtashami, S. Nousala, Inter-organizational Collaboration: Product, Knowledge and Risk, Journal of Systemics, Cybernetics, and Informatics, Vol. 9 (5), 30-35, Special Issue on Collaborative Enterprise, December 2011.

[16] N. Jastroch, V. Kirova, T. Marlowe, M. Mohtashami, Dams, Flows and Views: Cross-Aspect Use of Knowledge in Collaborative Software Development, Journal of Systemics, Cybernetics, and Informat-

ics, Vol. 9 (5), 36-40, Special Issue on Collaborative Enterprise, December 2011.

[17] V. Kirova, T. Marlowe, Prendre en compte les changements dynamiques dans le développement cooperative du logiciel, Génie Logiciel, Decembre 2008, Numéro 87, pages 15-25.

[18] G.L. Kolfschoten, P.B. Lowry, D.L. Dean, M. Kamal: A measurement framework for patterns of collaboration. HICCS Working paper, May 2007. http://www.hicss.hawaii.edu/Reports/41CEWorkshop.pdf (accessed January 2012).

[19] F. Koussouris, F. Lampathaki, S. Mouzakitis, Y. Charalabidis, J. Psarras: Digging into real-life enterprise interoperability areas - definition and overview of the main research areas. Contributed to the CENT 2011 Symposium, in: Proceedings of the 15th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2011), Vol. II, Orlando FL, July 2011.

[20] C. S. Ku, T. J. Marlowe: Software metrics for collaborative software engineering projects. Invited Session on Collaborative Knowledge Management (CKM 2010), Proceedings of the 4th International Conference on Knowledge Generation, Communication and Management (KGCM2010), June-July 2010.

[21] R. Laddad, Aspects in action: Practical aspect-oriented programming, Manning, 2003.

[22] C. Lange, M.Chaudron: Effects of Defects in UML Models – An Experimental Investigation. Proceedings of the 28th International Conference on Software Engineering, 20-28 May, 2006, Shanghai, China.

[23] C. Larman and B. Vodde, Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum, Addison-Wesley Professional, 2008.

[24] C. Larman and B. Vodde, Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum, Addison-Wesley Professional, 2010.

[25] D. Leffingwell, Agile Product Manager in the Enterprise (2): A Contemporary Framework, The Blog: Best Practices for Large Enterprises, May 19, 2009, http://scalingsoftwareagility.wordpress.com/2009/05/19/agile-product-manager-in-the-enterprise-2-a-contemporary-framework/

[26] K. MacIver, The Agile Revolution, 14 July, 2008, http://www.information-age.com/channels/development-and-integration/features/451431/the-agile-revolution.thtml

[27] T. J. Marlowe, V. Kirova, Addressing Change in Collaborative Software Development through Agility and Automated Traceability, WMSCI 2008, 209–215, Orlando, USA, June-July 2008.

[28] T. J. Marlowe, V. Kirova, High-level Component Interfaces for Collaborative Development: A Proposal, Journal of Systemics, Cybernetics, and Informatics, 7 (6), pages 1-6, 2009.

[29] T.J. Marlowe, N. Jastroch, V. Kirova, M. Mohtashami, A Classification of Collaborative Knowledge, Journal of Systemics, Cybernetics, and Informatics, Vol. 9 (7), 2011.

[30] T.J. Marlowe, N. Jastroch, S. Nousala, V. Kirova, The Collaborative Future, invited summary article, Journal of Systemics, Cybernetics, and Informatics, Vol. 9 (5), 1-5, Special Issue on Collaborative Enterprise, December 2011.

[31] T.J. Marlowe, N. Jastroch, S. Nousala, V. Kirova, Complex Collaboration, Knowledge Sharing and Interoperability, submitted to ICE2012.

[32] T.J. Marlowe, N. Jastroch, V. Kirova, An Approach for Discovery and Handling of Exceptions in Inter-Organizational Collaborative Software Development, submitted to Workshop on Exceptions and Exception Handling, International Conference on Software Engineering, 2012.

[33] M. Mohtashami, T. Marlowe, V. Kirova, F. Deek, Risk Management for Collaborative Software Development, Information Systems Management, 25 (4), 20–30, Fall 2006.

[34] M. Mohtashami, T. Marlowe, V. Kirova, F. P. Deek, A Comparison of Three Modes of Collaboration, 15th Americas Conference on Information Systems (AMCIS 2009) [CD-ROM], August 2009.

[35] M. Mohtashami, T. Marlowe, V. Kirova, F. Deek, Risk-Driven Management Contingency Policies in Collaborative Software Development, 40th Annual Meeting of the Design Sciences Institute (DSI 2009) [CD-ROM], New Orleans LA, November 2009.

[36] M. Mohtashami, C. Ku, T. Marlowe, Metrics Are Needed For Collaborative Software Development, Journal of Systemics, Cybernetics, and Informatics, Vol. 9 (5), 41-47, Special Issue on Collaborative Enterprise, December 2011.

[37] M. Mohtashami, T. Marlowe, V. Kirova, F. Deek, Risk-Driven Management Contingency Policies in Collaborative Software Development, International Journal of Information Technology and Management, Volume 10 (2-4), p 247-271, 2011.

[38] N. Nagappan, E. M. Maximilien, T. Bhat, L. Williams, Realizing quality improvement through test driven development: results and experiences of four industrial teams, Empirical Software Eng., 13:289–302, 2008.

[39] J. Pollock, R. Hodgson, Adaptive Information. Wiley-Interscience, 2004.

[40] N. Schadewitz, Cross-Cultural Collaboration, http://crossculturalcollaboration.pbworks.com/FrontPage

[41] K. Sullivan: Adaptation Architectures. Proceedings of the Third International Conference on Design Science Research in Information Systems and Technology. (V. Vaishnavi & R. Baskerville, Eds). May 7-9, 2008, Atlanta, Georgia: Georgia State University.

[42] G. Teichmann, E.M. Schwartz, F.M. Dittes: Collaborative Engineering of Inter-Enterprise Business Processes. Special Issue on Collaborative Enterprise, Journal of Systemics, Cybernetics, and Informatics (JSCI), Vol. 9 (5), p. 57 – 64, December 2011.

[43] J. Whitehead, Collaboration in Software Engineering: A Roadmap, 2007 Future of Software Engineering (FOSE'07), Minneapolis, Minnesota, May 23-25, 2007.