Do Agile GSD Experience Reports Help the Practitioner?

Philip S. Taylor, Des Greer, Paul Sage
Queen's University Belfast
Belfast BT7 1NN
Northern Ireland, UK
+44 (0)28 9097 4773
{p.taylor, des.greer, p.sage}@qub.ac.uk

Gerry Coleman, Kevin McDaid, Frank Keenan
Dundalk Institute of Technology
Dublin Road, Dundalk,
Co. Louth, Ireland
+353 42 9370200
{gerry.coleman, Kevin.mcdaid, frank.keenan}@dkit.ie

ABSTRACT
Agile software development has steadily gained momentum and acceptability as a viable approach to software development. As software development continues to take advantage of the global market, agile methods are also being attempted in geographically distributed settings. In this paper, the authors discuss the usefulness of published research on agile global software development for the practitioner. It is contended that such published work is of minimal value to the practitioner and does not add anything to the guidance available before the existence of current agile methods. A survey of agile GSD related publications, from XP/Agile conferences between 2001 and 2005, is used to support this claim. The paper ends with a number of proposals which aim to improve the usefulness of future agile GSD research and experience.

Categories and Subject Descriptors
D.2.9 [**Management**]: Software process models, programming teams.

General Terms
Experimentation, Theory.

Keywords
Agile Methods, Global Software Development, Experience Reports.

1. INTRODUCTION
Agile methods have become a viable option in the last five years for many software development organisations in numerous product domains. The literature related to agility in general and specific agile methods is voluminous given its recent origins. The presence of standard texts is coupled by an increasing amount of research papers presented at conferences such as the International Conference on eXtreme Programming and Agile Processes in Software Engineering and the Agile International Conference, and in journals such as IEEE Software [13] and Crosstalk [8].

This paper focuses on the research presented at the XP/Agile conferences from 2001 to 2005 relating specifically to agile global software development (GSD). The aim is to show that the research presented, almost entirely of an experiential nature, is of minimal value to the agile GSD practitioner or those about to become such practitioners.

This paper is organised as follows. The second section provides a short historical context for agile methods by discussing their evolution. The third section presents some of the drivers for GSD and outlines the guidance for practicing GSD from three important texts. Section 4 assesses the contribution of agile GSD experience reports from the XP/Agile conferences and Section 5 highlights the assumptions present in these experience reports. The sixth section discusses the usefulness of the agile GSD experience reports and Section 7 presents concluding thoughts and proposals for the future of agile GSD research.

2. AGILE METHODS IN CONTEXT
This section will briefly present a historical evolution of agile methods and thereby counter some of the misunderstandings that software organisations may have regarding their validity in the marketplace of software processes. For overviews of individual agile methods the reader can consult Abrahamsson et al [1] and Highsmith [12].

Larman and Basili [20, 21] have carefully provided the context for current agile methods. They argue convincingly that many of the practices which appear to be novel in agile methods, most notably incremental and iterative development (IID), have actually been practiced since software began to be developed in the 1950's.

When software began to be developed there were two approaches, IID and ad hoc. The waterfall process [29] was developed to improve those ad hoc development efforts and not necessarily to replace IID. The original waterfall approach is nuanced and Royce expects iteration between each stage and even supports early product release and close customer involvement. At some point a crude version of the waterfall process became the dominant approach, possibly due to its conceptual simplicity, and was used on many projects which would have been better suited to Royce's fully nuanced waterfall approach or IID. The misuse of the waterfall approach began to be readdressed in the early to mid

GSD'06, May 23, 2006, Shanghai, China.

1990's, borrowing many IID practices, resulting in what would later be known as Agile Methods.

Agile methods received their impetus from the short comings of heavily planned processes and crude waterfall processes to be successful with all varieties of software product and team and are now in the IID family. Agile methods are not ad hoc and their empirical nature requires discipline on the part of the team using them.

3. GSD PREDATING AGILE METHODS

Given the context for agile methods it is clear that GSD, in some form, will have been practiced before agile methods came into existence and gained popularity. It should also follow that agile GSD research and practice would build upon prior research and experience. This section surveys some of the ideas regarding GSD from the 1990's.

3.1 Drivers for GSD

GSD can be defined as any aspect of software engineering that involves the combined efforts of software professionals in different locations separated by significant distances. The potential for GSD is drawn from a number of business trends, two of the most important being globalisation and outsourcing, which can be witnessed readily in most established-economy countries and some emerging-economy countries.

Karolak [15] states that global software development is inevitable because of both "industry drivers, such as the supply and demand of technical resources, and the increasingly global software market, and business arrangements, such as strategic partnerships, joint ventures, and global companies" (p. 10).

Outsourcing is not necessarily cross-continent but involves software organisations engaging third parties in their development effort. Links are forged nationally as well as internationally. Meadows [27] highlights six issues, which are applicable to all forms of GSD, making outsourcing increasingly viable:

- Increased move toward component-based software development. Components of a system can be developed off-site with more ease.
- Increased standardisation. Programming languages and methodologies are not limited by culture and country therefore reducing the learning curve for software/hardware investments.
- Open systems adoption. Multiple development environments can be used.
- Reduction in mainframe environments. Increasing use of client/server environments reduces the capital investment requirements in emerging-economy countries.
- Improved communications. Transporting software engineering artefacts and bringing geographically distributed developers together is not an insurmountable problem, steadily becoming more cost effective and efficient.
- Rise of Integrated Project Support Environments (IPSE) providing a co-ordinated set of software engineering and management tools.

These issues are more prevalent within the current state of software engineering indicating that GSD is yet more viable than in the mid 1990's.

3.2 GSD Guidance

This sub-section briefly summarises the contribution of three texts to the area of GSD. The texts are Loftus et al [23], McConnell [26], and Karolak [15] released in 1995, 1996, and 1998 respectively. McConnell is a general project management text whilst Loftus and Karolak deal specifically with GSD. These texts represent a broad spread in emphases. Loftus et al [23] focus on detailed environment support for GSD whilst Karolak [15] provides specific business structure advice. McConnell [26] presents general high-level guidance.

As noted, Loftus et al [23] focus primarily on the tool support for GSD and advise that the environment to support collaborative software engineering must:

- Facilitate the sharing of project specific data.
- Hide confidential data.
- Address the problem of representing the same data in different environments.
- Not demand radical reorganisation of existing support environments.
- Be open to support the addition and removal of other development tools and environments.

The eventual process recommended by Loftus et al. [23] when practising GSD is as follows:

- Establish the nature of the relationships between the collaborators. Outline the overall structure of the project, in terms of activities, participants and channels of communication.
- Analyse the technologies the organisations are bringing to the project and decide how they fit together.
- Specify requirements for data sharing between collaborators, in terms of the support tools, files, databases, and so on, which will be used by the collaborators.
- Define logical software engineering environments within which project data will be shared.
- Document the actual architectures and data models which will be used.
- For each logical software engineering environment, specify the common data model which defines the data shared. Implement the data models in the actual environments on which the logical software engineering environment is to be built.

The work by Loftus et al [23] is fundamentally sound but with regard to tool requirements is somewhat dated today. The rapid improvement in Internet technologies has resulted in many of their issues being essentially solved.

McConnell [26] recommends outsourcing as a best practice for rapid development providing numerous reasons for its time-saving potential including: staffing flexibility; experience and expertise; contractually driven requirements specification; and reduced feature creep. Addressing offshore outsourcing specifically, McConnell [26] suggests several issues to keep in mind:

- Communication issues including reliable phone networks and language differences.
- Time differences can be a help or hindrance but, in any case, ensure that there is at least some overlapping time and a reliable e-mail system.
- Travel will be necessary at each major milestone of the outsourced development effort.
- Cultural, political and business characteristics of the country to which development is being outsourced must be understood

McConnell [26] lists five common risks and organisation-wide consequences of outsourcing many of which are also summarised by Aubert et al. [2]:

- Loss of visibility. Outsourcing can mean difficulty for tracking development progress and it is imperative that the contract stipulates when progress should be assessed and reported.
- Transfer of expertise outside your organisation. Two things can result from this feature of outsourcing: (1) your ability to develop 'in-house' the same software diminishes; and (2) the vendor's knowledge of your data and algorithms increases. An organisation should ask itself if the work being outsourced is part of its core business and competency [10]. If it is, outsourcing might be expedient in the short term, but it may reduce your competitiveness in the long-term.
- Loss of morale. Ensure that in-house developers are not under the impression that their own jobs are at risk or that they will never get included in interesting or challenging pieces of development at which they can increase their own knowledge and experience.
- Loss of control over future programs. You may lose the ability to extend the development in future as the vendor might make design and implementation decisions that limit future flexibility. Your developers are also unfamiliar with software developed outside their immediate departments. Again an appropriate contract is a necessity.
- Compromise of confidential information. Be sure to identify proprietary data and algorithms and ensure that this intellectual property stays carefully protected.

Karolak [15] views responsibility and accountability to be crucial and maintains that careful consideration can avoid many of the major problems associated with global software development. Responsibility is defined as the act of performing a task and the resulting actions. Accountability is accepting ownership of the activity regardless of who performed the tasks. Karolak's guidance for GSD is summarised in the following list:

- *Finalise business arrangement*. Identify and agree on the type and structure of the business arrangement used in developing the software, such as joint venture or strategic partnership.
- *Identify GSD team*. Identify the structure, members, member roles, and responsibilities of the team.
- *Identify GSD technology*. Identify the technical infrastructure which team members will use to communicate with each other.
- *Define statement of work*. Create the document used to identify the software development responsibilities and expectations between the customer and supplier.
- *Divide the work.* Divide the effort among software developers by staffing, business relationship, expertise level, and so forth.
- *Identify tools and methods*. Identify the software development tools and design and development methods.
- *Establish virtual software configuration control board (SCCB)*. Identify the members of the SCCB, the method they will use for software configuration management, and how frequently they will meet.
- *Identify and manage risks*. Identify risks and devise a risk mitigation strategy for each class of risk.
- *Control documentation*. Identify a control method and apply it to all project documentation.
- *Develop and apply test suites*. Identify test suites during software design and code development. During testing, perform software verification and validation using test suites.
- *Develop and apply a traceability matrix*. Create the matrix during requirements and update it during design and code generation.
- *Develop and apply a module version matrix*. Identifies a module or component to the configuration version it uses in a software build.
- *Establish maintenance review board*. Its purpose is to review requests for changes after the product has been delivered to the customer.
- *Control software quality*. Perform activities that enhance the quality of software and ensure that it meets the customer's expectations.
- *Manage intellectual property*. Perform activities, such as design reviews, to determine if ideas generated during development should be protected as intellectual property.

Although these texts have arisen from the context of software development as practiced in the 1980's and 1990's, the general guidance is applicable to numerous current software development contexts including agile GSD. The guidance provided ranges from general to specific with regard to both business practice and tool support. An agile GSD practitioner could easily use the guidance from these texts even though they were not written from an agile methods context.

4. AGILE GSD RESEARCH PAPERS ASSESSED

This section discusses the papers presented at the XP/Agile conferences, from 2001 to 2005, addressing some aspect of agile GSD. Summary categories are as follows:

- *Industrial Experience* - Is the paper an industrial experience report?
- *Experimental* - Does the paper present experimental research?
- *Tools* - Does the paper present new tools for agile GSD?
- *Process Emphasis* - Does the paper seriously address agile software processes?
- *Useful Practices* - Does the paper present useful practices?

From a total of fourteen papers, eight are industrial experience reports and seven present the results of experimental research. Seven of the papers deal with tool support for agile GSD and ten present useful practices for agile GSD. Four out of fourteen deal specifically with distributed pair programming and do not present other useful practices for agile GSD. None of the papers deal adequately with process improvement issues for agile GSD.

None of the industrial experience reports make reference to the GSD related texts summarised in Section 3.2. Only three of the seven industrial experience reports [6, 24, and 14] make reference to any GSD related texts [7, 19, and 22] and do not interact seriously with them. Braithwaite and Joyce [6] actually chose not to investigate the GSD literature before experimenting with distributed Extreme Programming, utilising the XP concept of courage out of context.

To assess the usefulness of the guidance presented in the experience reports the existing GSD guidance presented in Section 3.2 was consolidated in one list of nineteen points. The GSD guidance was then extracted from the eight industrial experience reports and compared to the existing GSD guidance to find any overlap. Table 1 (see Appendix), as an example of the assessment method, contains a comparison using Kircher et al [18] and Danait [9]. These experience reports are separated by four years and Kircher et al [18] deals specifically with XP while Danait [9] addresses general agile development. It is clear that the agile GSD experience reports do not add much guidance that is not already in existence from standard GSD texts. Much use is made of general GSD supporting technologies and the importance of cross-location visits is emphasised.

When the agile GSD experience reports present a new practice it is often not feasible. An example of distributed XP guidance presented by Braithwaite and Joyce [6], typical of the papers reviewed, is provided in Table 2 (see Appendix). They wisely advocate balanced sites but their solution – make all sites equal in skill and numbers – is impossible in many agile GSD scenarios.

5. ASSUMPTIONS OF AGILE GSD EXPERIENCE REPORTS

It is important to carefully assess these experience reports and isolate the assumptions inherent within them before their usefulness can be commented on.

First, it appears that the agile GSD experience reports assume the novelty of agile methods. However, as presented in Section 2, agile methods have a clear historical context which contains practical guidance for GSD. Section 3.2 contains a wealth of guidance from only three texts. Lack of interaction with such practical guidance as embodied in the general texts and research publications is presumptuous and not typical of thorough research methods.

Second, it is implicitly assumed that specific agile methods are fixed whether being used in co-located development or GSD. It is a valuable research activity to assume the fixed nature of agile methods and explore their strengths and weaknesses for GSD. However, such a research activity does not need to be repeated continuously in experience reports with each yielding similar results.

Third, there is often the assumption that specific agile method practices such as pair programming or daily face-to-face meetings must be replicated (often with webcams) in a geographically distributed setting. Again, although this is a useful research activity it is question begging. More thought needs to be given to the pre-agile GSD practices and how their inherent discipline might be a better risk balance to agile practices when performing GSD. This is the approach taken by Boehm and Turner [3, 4, and 5].

Fourth, there is the assumption of development context. The agile GSD experience reports reviewed do not discuss the nature of the distribution, product domain, and contract arrangements. Unless such context details are discussed it is nearly impossible for the practitioner to seriously interact with the guidance provided in the experience reports. Such issues are still the topic of research in co-located agile development and present even greater risks to agile GSD.

Fifth, there is an implicit assumption that agile methods are mature and without criticism. It is noticeable that the agile GSD experience reports do not interact with the texts and papers that highlight perceived weaknesses in agile methods. Turk et al [30, 31] have discussed some of the problems they perceive with agile methods. Their work is based primarily on examining the underlying assumptions of agile methods and determining for which development scenarios the assumptions do not hold. They arrive at two groups of limitations:

Personnel limitations

- Limited support for distributed development environments
- Limited support for subcontracting
- Limited support for large teams

Product limitations

- Limited support for building reusable artifacts
- Limited support for developing safety-critical software
- Limited support for developing large, complex software

Such limitations are useful pointers to areas that will prove difficult when practicing agile GSD. For example, developing safety-critical software is an area of research for co-located agile development but the risks will be magnified for agile GSD. Other studies by Keefer [16] and McBreen [25] focus specifically on perceived weaknesses with Extreme Programming (XP) [17]. They also note similar limitations to Turk et al [30, 31].

6. USEFULNESS OF AGILE GSD EXPERIENCE REPORTS

Given the assumptions presented in Section 5 it is the position of this paper that agile GSD experience reports are of minimal value to the practitioner. Those involved in, or about to begin agile GSD would be better advised to examine the texts and research relating to general GSD and then carefully examine their specific development context to discover the major risks.

The future for agile GSD research needs to essentially follow and build upon the careful research and experience presented in such forums as the ICSE Workshop on GSD and the ICSE conferences in general. A good example of the potential of experience reports can be found in Herbsleb et al [11]. They clearly present their

research method, context and resulting advice. Although the advice does not contribute anything new to the field of GSD they have at least been more rigorous in their research. Paasivaara and Lassenius [28] present research and guidance related to the practice of IID in GSD scenarios. This paper is also a good example of an experience report providing a good overview of context and research methodology.

It could be argued that the purpose of an experience report should simply be to outline an experience irrespective of whether such an experience and the resulting guidance has been documented before. Such a view would not require any significant research on the part of the report writers but would also mean the experience report would require careful contextualizing by both the practitioner and the researcher. However, what needs to be avoided is the 'reinventing the wheel' syndrome when each new experience report published simply presents the same resulting guidance for agile GSD.

The 2004 ICSE workshop on GSD [28] had the following emphases: feasibility of GSD; strategies for success of GSD; research methods and challenges in GSD. Each of these emphases needs to be applied to agile GSD research and practice.

The following issues emerged from the ICSE workshop as a whole:

- Increased community building.
- More systematic application and documentation of research methods.
- Building defined models and theories.
- Defining the state of the practice of software engineering.

All but the first of these emerging issues require further work in agile GSD research:

Increased community building. On this issue the agile practitioners and researchers are effective. There are numerous, perhaps too numerous, agile community websites and e-mail lists that distribute the latest thought on agile methods. The authors of this paper have instigated such community building in Ireland by organising and hosting agile events. A budget is set aside for promotional material and visiting speakers. Many other groups in numerous countries are doing the same activity.

More systematic application and documentation of research methods. The XP/Agile conferences are beginning to show that more serious thought is being given to research methods. However, there is much research that relies on assumptions and does not give due care to development context and historical GSD practice and guidance.

Building defined models and theories. On this issue the agile research is still relatively weak and it could be argued that defined models and theories are antithetical to the principles of agile methods. However, if agile research is to be useful to the broadest range of practitioners it must have some rigor and thoroughness.

Defining the state of the practice of software engineering. A state of practice survey has yet to be completed with reference to agile GSD. In some sense the agile GSD experience reports are documenting a state of practice but more needs to be done to consolidate such experience reports and to carefully analyse the assumptions underlying them.

7. CONCLUSIONS

This paper has aimed to show that agile GSD experience reports, as published in the proceedings of the XP/Agile conferences, are of minimal value to the agile GSD practitioner. The experience reports surveyed were based on some inherent assumptions with regard to: the historical context of agile methods; the evolving nature of agile methods; the practices of agile methods; the uniformity of development context; and the maturity of agile methods.

It has been argued that the agile GSD guidance provided in the experience reports does not add anything to the existing guidance contained in a sampling of published textbooks. In effect, the experience reports are reinventing the wheel with regard to agile GSD practice.

In light of the survey and discussion presented in the previous sections, the future direction and relevance of agile GSD research and practice will be strengthened through the following efforts:

- *The placing of agile GSD within the context of GSD*. There are benefits to specialised groups within the software engineering community but not to the extent that the specialised groups exclude the research contribution of each other. Agile GSD seems to have suffered in this regard and therefore it is proposed that the best agile GSD experience based research be primarily part of the ICSE Workshop on GSD for the Practitioner. Research presented in this workshop should address the following points.

- *Study the feasibility of agile GSD*. Those with agile experience and those with GSD experience need to work together to establish the feasibility of agile GSD. It needs to be determined if the same benefits of a co-located agile approach can actually be achieved with agile GSD. It is proposed that more careful integration with business management research related to outsourcing and globalization will help to provide a foundation for the feasibility of agile GSD.

- *Capturing the state of practice*. The practitioners and researchers of agile GSD need to collaborate and arrive at a state of practice. It is proposed that agile GSD practitioners and researchers from each continent begin to compile and publish such a state of practice which can inform future agile GSD research.

- *Research methods and challenges*. Based on the state of practice the research methods need to be clearer with better defined models and theories. The challenges also need to be collated and used as a foundation for further improvements in practice research. It is proposed that conference workshops be the first place to discuss the methods and challenges.

- *Increased community building within agile GSD and with other expertise.* Business researchers can help inform on issues such as contractual arrangements, revenue generation, strategic partnerships and so forth. The human-computer interaction community can provide valuable guidance with regard to tool support for agile GSD. It is proposed that an ontology of expertise that can inform agile GSD challenges be developed and presented in a future workshop.

It is believed that implementing the above proposals will provide a useful foundation for future agile GSD research.

8. REFERENCES

[1] Abrahamsson, P., Warsta, J., Siponen, M. T., Ronkainen, J. New Directions On Agile Methods: A Comparative Analysis. *Proc. 25th Int. Conf. Software Engineering*. IEEE Computer Society (2003) 244 – 254

[2] Aubert, B. A, Dussault, S., Patry, M., and Rivard, S. Managing the Risk of IT Outsourcing. *32nd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society (Digital Library Edition) 1999

[3] Boehm, B., Turner, R. Rebalancing Your Organization's Discipline and Agility. In: Maurer, F., Wells, D (eds.): *XP/Agile Universe 2003*. Springer-Verlag, Berlin Heidelberg (2003) 1 – 8

[4] Boehm, B., Turner, R. Using Risk to Balance Agile and Plan-Driven Methods. *IEEE Computer*, Vol. 36(6), IEEE Computer Society (2003) 57 – 66

[5] Boehm, B., Turner, R. *Balancing Agility and Discipline – A Guide for the Perplexed*. Addison-Wesley (2004)

[6] Braithwaite, K. and Joyce, T. XP Expanded: Distributed Extreme Programming. *6th International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Springer, 2005, pp. 180-188

[7] Carmel, E. *Global Software Teams: Collaborating Across Borders and Time Zones*. Prentice Hall (1998).

[8] *Crosstalk: The Journal of Defense Software Engineering*. Vol. 15(10), 2002, special issue on Agile Software Development.

[9] Danait, A. Agile Offshore Techniques – A Case Study, Agile 2005, *available at http://www.agile2005.org/XR17.pdf* (last visited March 2006)

[10] Hancox, M. and Hackney, R. Information Technology Outsourcing: Conceptualising Practice in the Public and Private Sector. *32nd Annual Hawaii International Conference on System Sciences*, IEEE Computer Society (Digital Library Edition) 1999.

[11] Herbsleb, J. D., Paulish, D. J., and Bass, M. Global Software Development at Siemens: Experience from Nine Projects. *26th International Conference on Software Engineering*, IEEE Computer Society, 2004, pp. 542 – 533

[12] Highsmith, J. *Agile Software Development Ecosystems*. Addison-Wesley (2002)

[13] *IEEE Software*. Vol. 18(6), 2001, special issue on Extreme Programming.

[14] Jensen, B. and Zilmer, A. Cross-Continent Development Using Scrum and XP. *4th International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Springer, 2003, pp. 146-153

[15] Karolak, D. W. *Global Software Development*. IEEE Computer Society Press (1998)

[16] Keefer, G. Extreme Programming Considered Harmful for Reliable Software Development 2.0. *Available at http://www.avoca-vsm.com/Dateien-Download/ExtremeProgramming.pdf* (last visited January 2006). AVOCA GmbH 2003

[17] Kent, B., Andres, C. *Extreme Programming Explained: Embrace Change*. 2nd Ed. Addison-Wesley (2005)

[18] Kircher, M., Jain, P., Corsaro, A., and Levine, D. Distributed eXtreme Programming, XP 2001, *available at http://www.kircher-schwanninger.de/michael/publications/xp2001.pdf* (last visited March 2006)

[19] Klepper, R. and Jones, W. O. *Outsourcing Information Technology and Services*. Prentice Hall (1997).

[20] Larman, C. *Agile & Iterative Development – A Manager's Guide*. Addison-Wesley (2004)

[21] Larman, C., Basili, V. R. Iterative and Incremental Development: A Brief History. *IEEE Computer*, Vol. 36(6), IEEE Computer Society (2003) 47 – 56

[22] Lipnack, J. and Stamps, J. *Virtual Teams: Reaching Across Space, Time, and Organizations with Technology*. John Wiley & Sons (1st Ed. 1997, 2nd Ed. 2000)

[23] Loftus, C. W., Sherratt, E. M., Gautier, R. J., Grandi, P. A. M., Price, D. E., and Tedd, M. D. *Distributed Software Engineering*. Prentice Hall (1995)

[24] Martin, A., Biddle, B., and Noble, J. When XP Met Outsourcing. *5th International Conference on eXtreme Programming and Agile Processes in Software Engineering*, Springer, 2004, pp. 51-59

[25] McBreen, P. *Questioning Extreme Programming*. Addison-Wesley (2003)

[26] McConnell, S. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press (1996)

[27] Meadows, C. J. Globalizing Software Development. *Journal of Global Information Management*, 4(1), 1996, pp. 5-14

[28] Paasivaara, M. and Lassenius, C. Using Iterative and Incremental Processes in Global Software Development. *3rd International Workshop on Global Software Development*, 2004, pp. 42-47 *available at http://gsd2004.uvic.ca/docs/proceedings.pdf* (last visited January 2006)

[29] Royce, W. W.: Managing the Development of Large Software Systems. *Proc. WESCON*. IEEE Computer Society (1970) 1 – 9. Available for download at http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf (last visited January 2006)

[30] Turk, D., France, R., Rumpe, B. Limitations of Agile Software Processes. In: Wells, D., Williams, L. A. (eds): *XP/Agile Universe 2002*. Springer-Verlag, Berlin Heidelberg (2002) 43 – 46

[31] Turk, D., France, R., Rumpe, B. Assumptions Underlying Agile Software Development Processes. *Journal of Database Management*, Vol. 16(4), Idea Group Inc (2005) 62 – 87

9. Appendix

Table 1. A Comparison of Existing GSD Guidance with Agile GSD Guidance

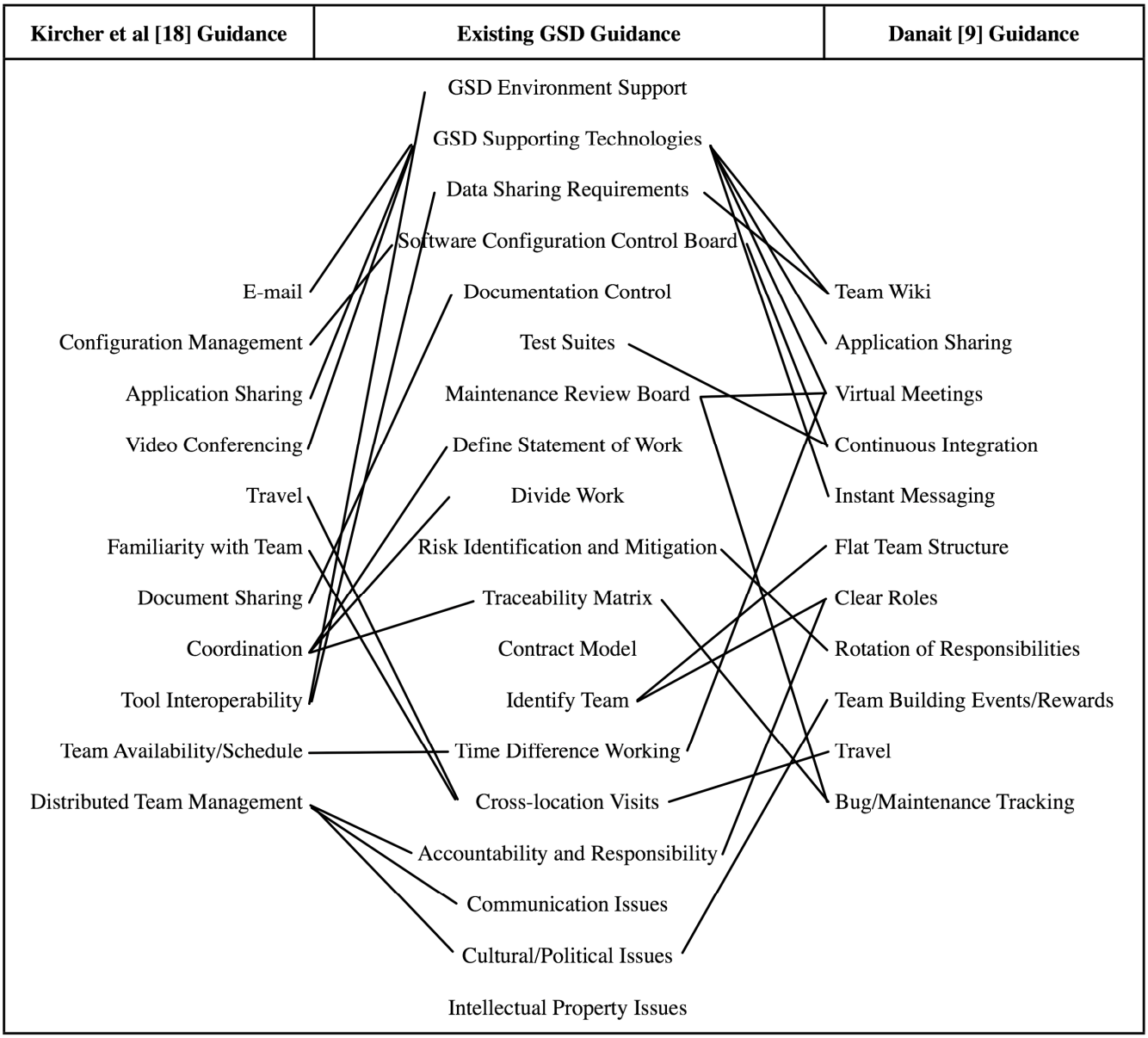

Table 2. Typical Example of the Problems and Solutions for Agile GSD

Practice	Problem and Solution
One team	*Problem* – Trust and cooperation between team members can break down.
	Solution – Maintain single team identity and encourage non-business communication.
Balanced sites	*Problem* – The difference in skill and experience can leave decisions to one particular team/site.
	Solution – Make all sites equal in skill and numbers.
Distributed standup	*Problem* – Remote team members cannot easily discuss project issues.
	Solution – Have a video conference session running whenever possible. Force an overlap if required.