A Systems Approach to Software Process Improvement in Small Organisations

Diana Kirk and Stephen G. MacDonell

SERL, Auckland University of Technology
Private Bag 92006, Auckland 1142, New Zealand
{dkirk, stephen.macdonell}@aut.ac.nz

Abstract

There is, at the present time, no model to effectively support context-aware process change in small software organisations. The assessment reference models, for example, SPICE and CMMI, provide a tool for identifying gaps with best practice, but do not take into account group culture and environment, and do not help with prioritisation. These approaches thus do not support the many small software organisations that need to make effective changes that are linked to business objectives in short time periods. In this paper, we propose a model to support such change. We base the model on an analogy of 'software system as human' and suggest that we can apply the idea of human health to help identify business objectives and improvement steps appropriate for these objectives. We describe a 'proof-of-concept' case study in which the model is retrospectively applied to a process improvement effort with a local software group.

Keywords: *Software process improvement, process modelling, systems approach*

1. INTRODUCTION

Existing models for software process improvement (SPI), for example, CMMI (Chrissis et al. 2007), ISO/IEC 15504 (SPICE) (ISO 2006) and ISO/IEC 12207 (ISO 1997), have been criticised by researchers and practitioners as being limited to large, traditional organisations and failing to provide the necessary guidance for small software groups (CaterSteel 2001, Grunbacher 1997, Huack et al. 2008, McCaffery et al. 2008). As software groups comprising fewer than 25 persons represent a majority in Europe, Ireland, Canada, Brazil and elsewhere (Laporte et al. 2008), this is seen as a major issue. Some characteristics of smaller organisations that may affect the success of SPI adoption include more informal management and planning, greater need for flexibility, a human-centric culture (Laporte et al. 2008), 'flatter' structure with imprecisely-defined responsibilities, a lack of exposure to standards and limited funds (Grunbacher 1997). Such characteristics imply an environment where dependence upon individuals is high. It has also been observed that existing models state which processes should be in place, but provide no guidance as to which to implement first (Chen et al. 2008). This means that the need of small organisations to prioritise according to business objectives (Aaen 2003, Laporte et al. 2008) is not supported.

We have earlier suggested that a model to support activity selection during software projects must take into account the business-related objectives for the software project and have proposed that a more holistic and flexible approach to process selection involves a change in focus from 'defining activities' to 'selecting activities to meet objectives' (Kirk 2007). In this approach, focus is on the whole system i.e. is not limited to considerations of cost and quality but rather includes consideration of human-related factors. For example, the owner of a small software organisation may be extremely interested in retaining and increasing the knowledge of developers and so may consider 'developer knowledge' to be of importance. This understanding may inform his choice of process and he may, for example, choose informal reviews over unit testing as a means of meeting objectives.

The need to focus on system objectives during software process improvement (SPI) initiatives is also suggested by others. Aaen (2003) criticises the use of existing assessment models as creating "a blueprint of a future software process" without providing any understanding of how processes emerge. He believes that it is necessary to understand an organisation's values and goals before understanding how it may change and that a preferred approach would be to support process users in deciding what a specific situation requires. Laporte et al. (2008) have found that one reason for the failure of small organisations to adopt standard models is that such organisations "find it difficult to relate ISO/IEC 12207 to their business needs".

We have suggested that a fruitful analogy to aid understanding of contextualised software systems is to consider the software system as a human and to apply ideas from human health (MacDonell et al. 2008). With this analogy, the focus is on identifying gaps in values of relevant 'health' factors and selecting activities to close the gaps. In this paper, we extend this analogy and apply it as a basis for a model for SPI. We explore the potential usefulness of the model by retrospectively applying it to an SPI initiative with a small, local software group. We then show how we have used the model as a basis for creating hypotheses for more formal investigation in small software organisations.

2. RELATED WORK

There is increasing interest in supporting software process improvement in small groups as a result of the realisation that such groups form a majority in many countries (Laporte et al. 2008). A selection from the literature is presented below.

The International Standards Organisation (ISO) has established a working group to address the creation of a software engineering standard tailored to very small enterprises (Laporte et al. 2008). The approach taken is to tailor an existing Mexican standard, *MoProsoft*, for small and medium enterprises. MoProsoft is based on ISO/IEC 12207, with practices from ISO9001, CMMI, the Project Management Body of Knowledge and the Software Engineering Body of Knowledge.

The University of Southern Queensland has developed a method, *RAPID* for software process assessment in small organisations and have applied this method in four organisations (CaterSteel 2001). The approach involved selecting eight processes based on ISO/IEC 15504 and restricting assessment to rating levels 1-3.

McCaffery et al. (2008) introduce *AHAA*, a "new low-overhead method that has been designed for small-to-medium-sized organisations wishing to be automative software suppliers". The method integrates the structuredness of CMMI and Automative SPICE with the flexibility of agile practices. The development of AHAA included a restriction of the CMMI process areas most suitable for inclusion in an SPI model for small-to-medium-sized organisations, based on a number of criteria extracted from the literature. The four process areas selected for the first release were Requirements Management, Project Planning, Project Monitoring and Control and Configuration Management.

Pikkarainen et al (2005) discuss deploying agile practices in organisations and applies a framework based on a continuous improvement ideology that "addresses the importance of utilizing the experiences of the software developers" as an important input to SPI. The approach involves selecting the agile practices to be deployed and the author comments that the "existing ways to discover the agile methods to deploy are unstructured" (Pikkarainen et al. 2005).

In the above examples, the approach is to select a subset of process areas from established models and create assessment models based on this subset. The resulting models have been applied with some success. However, none supports the ability to choose a project-specific development model based upon key objectives or to make tradeoffs when planning changes (MacCormack et al. 2003).

3. HUMAN HEALTH ANALOGY

We have proposed that a useful analogy to aid understanding of software system health is that of the human system. In this Section, we expand on this analogy in order to provide a motivation for our SPI model. Some drivers of the analogy are (MacDonell et al. 2008):

- Human health is established by measurement of indicators, for example, blood pressure and cholesterol levels. We measure the 'health' of a software system (in its broadest sense, as described in Section 4) by indicators such as cost and defect levels and stakeholder satisfaction.
- Humans pass through a number of life stages, for example, adolescence and mid-life crisis. Each stage exhibits some common characteristics. For example, midlife crisis might occur when the children leave home and 'business as usual' is no longer appropriate, forcing a struggle to fit in with new situations and

expectations. Software systems may also be perceived as having similar 'life stages'. For example, a step change in technology may result in an established software product no longer behaving as required, forcing efforts to make the product 'fit in'.

- The relevant indicators for humans and their expected values depend upon the life stage. For example, an Apgar test is carried out on newborn babies to establish health; the 'normal' pulse rate for an infant is different from the 'normal' rate for an adult. In a similar way, for a software system it is expected that the numbers of defects identified when the system is 'in adulthood' (i.e. established in the field) will be far fewer than when the system is 'in embryo' (under development).
- Human health is dependent upon environmental factors. For example, a thin person may be 'healthy' in a hot country with food freely available but may not fare so well in a very cold climate with low food availability. In a similar way, software targeted for experienced users may cease to be 'healthy' when the customer base extends to include naive users.
- Human health can be affected by behaviours. For example, mothers can support a positive outcome for babies by eating well and not smoking. Software systems can also be affected by behaviours. For example, developers can support a positive outcome by following best practices.
- Once a human becomes unhealthy (as defined by indicators such as blood pressure), considerable effort is required to return to health. Success depends upon the human's willingness to change behaviours and the availability of opportunities to effect the new behaviours. For 'unhealthy' software systems, considerable effort is also required to effect change as factors such as cost and resistance-to-change come into effect.

We observe that a human may embark upon a health improvement initiative for one of a number of reasons (MacDonell et al. 2008):

- Sickness. The person may be experiencing symptoms that indicate sickness, for example, chest pains or headaches. The physician will probably check a number of key indicators, for example, blood pressure and temperature, for values that deviate from 'normal'. As a result of findings, the physician will infer the root cause of the symptoms and suggest a treatment that will remove the root cause, thus returning indicator values to 'normal' and removing symptoms. During diagnosis of root cause, the physician will probably take into account the specific life stage of the person. The suggested treatment must a) take into account the human system in a holistic way and b) consider the constraints imposed by contexts. For example, medication that lowers blood pressure but induces depression is probably not an ideal solution; nor is medication that lowers blood pressure for a person who reliably fails to take prescribed medication.
- Prevention. The person may choose to monitor health in a proactive way, for example, undergo a yearly check of cholesterol levels and blood pressure. Should values be abnormal, the physician will generally progress as for 'sickness'.
- Growth. The person may have some goal that involves improving physical or mental capability, for example, 'run a marathon'. In this case, the first task is to identify appropriate indicators and the changes required, for example, 'increase stamina' and the next task is to choose a suitable activity that addresses required changes, for example, 'running'. Again, choosing a suitable activity involves both considering indicators in a holistic way and identifying factors that may affect success. For example, if I live on a busy street and lack motivation, I may decide that 'personal trainer at the gym' will give me a better chance of success than 'running a circuit from my home at 5:30 a.m.'. However, if I am concerned about financial status, a personal trainer might be too expensive and I might decide to join a group fitness class instead.
- Adaptation. The person may be required to move to a new environment, for example, leave the childhood home or move to a different country. Behaviours that worked well in the original environment, for example, leaving cooking to others or speaking in English, may be ineffective in the new one. To mitigate the risk of failure-to-adapt, (s)he must identify the gap in key indicators (for example, `independence') and aim to close the gap by suitable activity selection.

In Figure 1, we illustrate the analogy with an example for each motivation (MacDonell et al. 2008). A key observation from the analogy is that, rather than focusing on processes, as is common for SPI models, we focus on goals and indicators and it is the indicator values that inform processes. Relevant indicators are situation-specific and thus appropriate process is situation-specific. For example, if I want to improve my ability to speak French, I do not need to improve my cholesterol level.

In the next Section, we introduce a model for SPI based on an extension of the above analogy.

Scenario	Person	Software
Sickness - Symptoms - Indicators unhealthy - Find cause and treat	Headache Blood pressure Take medicine	Customers unhappy Defect numbers Requirements process
Prevention - Monitor indicators - Preventative action	Cholesterol, lipids Lifestyle change	Defect levels Process/product change
Growth - Identify objectives - Confounding factors - Make changes	Run a marathon Motivation Training, diet	New innovative product Processes don't support creativity Gap analysis and change
Adaptation - New environment - Indicators gaps - Confounding factors - Close gaps	Redundancy Computer skills Confidence Computer course	Business environment All web-based No web expertise Hire web developers

Figure 1: Human and software systems: SPI examples

4. PROPOSED MODEL

We commence our description of the proposed model by overviewing the architecture of a software system from the perspective of our analogy. A software system comprises a number of components (see Figure 2) and associated with each of these is a number of representative characteristics.

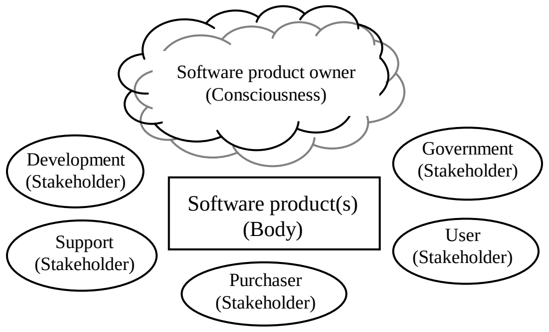

Figure 2: Software system components

- *Software product.* The 'body' of the software system is the software product or products. Common characteristics include quality indicators, such as defect density, cost indicators, such as effort, and content indicators, such as number of features.

- *Software product owner.* The 'consciousness' of the software system is represented by the entity that has authority for making decisions about planned change to the *software product* or its *stakeholders*, generally the organisation responsible for creating and deploying the software product(s). Common characteristics include organisational maturity, size, culture and management style.

- *Stakeholders.* The environment for the software system includes all humans with an interest in the *software product*. These may include members of the development organisation (for example, developers, project management, QA and support personnel) and the deployment organisation (for example, purchasers and users). Stakeholders may effect unplanned change to the product environment. For example, experienced users of a product may be replaced by inexperienced users. Common characteristics relate to skills, experience and personality type.

We now extend the ideas from Section 3 to create a methodology for analysing the health of software organisations and recommending change. We begin with some definitions:

- **Software system**. Comprises the *software product*, the *software product owner* and *stakeholders*.

- **Key indicators**. Factors that characterise the various components of the *software system* and are identified as being relevant for a specific SPI initiative.

- **Goal indicators.** *Key indicators* that represent the desired level of health of a software system, for example, relating to cost, quality and satisfaction levels.

- **Context indicators**. *Key indicators* that characterise the software system's ability to change the values of *goal indicators*, for example, relating to cost, motivation and skill levels.
- **Software system lifecycle**. The stages through which a software system passes, for example, 'Childhood' and 'Adolescence', as defined in MacDonell et al. (2008). Each stage is associated with changes to some *key indicators*, for example, 'adolescence' is associated with high levels of defects discovered in the field.
- **Symptom.** Problem reported by any *stakeholder*.
- **Prevention.** An assessment requested by the *software product owner* in which no *symptoms* are reported, rather the need is to 'check that everything is fine'. The assessment will result in a categorisation of the *software system* as one of *sickness*, *growth*, *adaptation* or *health*.
- **Growth.** Planned change to *product* or *product owner* that is outside 'business as usual', resulting in a gap between current and desired *goal indicator* values. For example, a plan for an innovative new product may mean that developer and test expertise becomes 'low'.
- **Adaptation.** Unplanned change to *stakeholders* also results in a gap between current and desired *goal indicator* values. For example, if naive users are permitted to use a product intended for use by experienced users, the 'product usability' level will fall.
- **Sickness.** Values of *goal indicators* are lower than expected for one or more of *software product*, *software product owner* or *stakeholders* and the *software system* is not in *growth* or *adaptation*. For example, effort or defect numbers may be too high or satisfaction levels too low.

Application of the model involves carrying out the following three steps (see Figure 3). For each, we present some examples to illustrate the need to take into account *context indicators* and *software system lifecycle* stages.

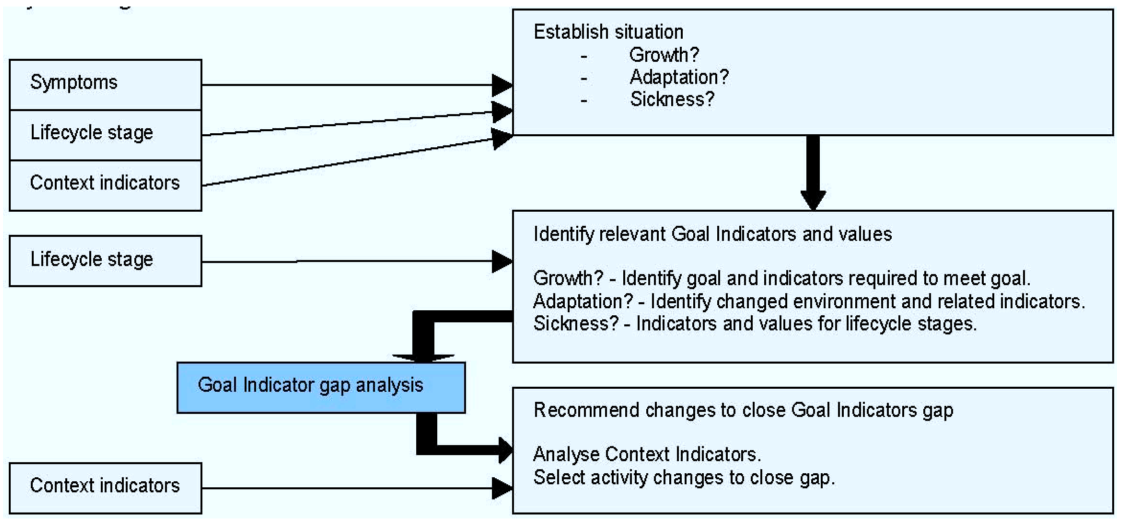
Symptoms
Lifecycle stage
Context indicators
Establish situation
- Growth?
- Adaptation?
- Sickness?
Lifecycle stage
Identify relevant Goal Indicators and values
Growth? - Identify goal and indicators required to meet goal.
Adaptation? - Identify changed environment and related indicators.
Sickness? - Indicators and values for lifecycle stages.
Goal Indicator gap analysis
Recommend changes to close Goal Indicators gap
Analyse Context Indicators.
Select activity changes to close gap.
Context indicators

Figure 3: SPI model steps

4.1 Step 1: Establish if growth, adaptation or sickness

We first interview staff to establish whether the initiative relates to a situation of *growth*, *adaptation* or *sickness*. We take this approach also in the case of a *preventative* assessment initiative.

For *growth*, we look for 'business-not-as-usual'. For example, a medium-sized organisation, A, is 'doing well', with a mature product sold to a global market (*adulthood*). Although management reports some existing problems with quality, we learn that there are plans to launch a new, innovative product into a marketplace characterised by rapidly changing technology. We categorise as a *growth* situation.

A mature organisation, *B*, with an established product used by experienced personnel would like an assessment to 'check things out' (*prevention*). When interviewing members of the support team, we discover that an increasing number of issues are being logged by users who 'do not know how to use the product'.

Further probing with management reveals that downsizing in the client sector has resulted in the product being used by ‘naive' users. The situation is one of *adaptation*.

Organisation C is a small group with low levels of formal process and reports problems of product quality (*symptoms*). In the absence of *growth* or *adaptation* scenarios, we categorise as *sickness*.

4.2 Step 2: Identify goal indicators and establish gap

We next establish the business objectives of interest within the given situation. We use these to help inform *goal indicators* and establish gaps between desired and current values.

Goals for organisation A relate to timely delivery and marketing and selected *goal indicators* are ‘time to market’ and ‘number of hits on web page’. Organisation B decides that, as the client base comprises a small number of large clients, it must focus on keeping existing clients happy. Selected *goal indicator* is ‘client satisfaction levels'. Goals for C relate to defect levels and the group decides to focus on ‘defects found during testing’.

4.3 Step 3: Choose activities to close gap

We finally work with the organisation to establish appropriate activities to close gaps between the current and desired values of *goal indicators*. To help inform choice, we consider relevant *context indicators*, existing standards such as ISO/IEC 12207 (ISO 1997) and the organisational literature.

Organisation A is structured into marketing, development and QA teams and has in place some sound development processes. We understand from the literature that “more flexible product development procedures are important to the success of new products in dynamic environments” (Carbonell & Rodriguez-Escudero, 2009, p. 32, citing Henard & Szymanski, 2001) and that innovation effectiveness is supported by the use of cross-functional teams in conjunction with strong management support (p. 29, citing Cooper & Edgett, 2008). We suggest that such a team be set up and supported by the owner.

For organisation B, possible activities to improve ‘client satisfaction levels’ may include upgrading the product, assigning a client advocate and weekly contact. We learn that a new version of the product is pending and management, now aware of the dangers of reduced satisfaction levels, chooses to assign a dedicated client advocate to each major client during the transition period.

For C, we identify *context indicators* and values as 'low process knowledge', 'culture flexible', 'no spare time' and ’motivated’. We also learn that the source of the problem is believed to be lack of clarity about the product requirements i.e. resides in the interface between product definition, development and QA. We decide that the most appropriate way to support process change is to provide options relating to the situation and work with group members to establish the most acceptable option(s). The group decides to hold a weekly meeting at which uncertainties in features will be identified and a senior member assigned to flesh out features, if deemed necessary.

5. CASE STUDY

In this Section, we describe how the model was retrospectively applied in the context of a software process improvement initiative in a small software organisation in Auckland. For reasons of confidentiality, only relevant aspects of the study are reported.

As is common for small organisations, members of the target team had an in-depth knowledge of the product and client base. Each member 'owned' one or more roles that included development with both existing and new technologies, testing and support for the client-facing sections of the organisation. Management was very happy with the group's performance and simply wanted to confirm that nothing important was being missed. Interviews aimed at understanding strategic objectives were held with the management team and individual interviews aimed at uncovering potential issues were held with group members. These were followed by two group sessions aimed at consolidating and agreeing on issues and brainstorming appropriate solutions.

The ISO/IEC 12207 model (ISO 1997) was applied in the backgound as reference model. However, the target team operated at a very immature level with respect to this model, with virtually no process areas formalised at any level. Regardless of this, the team appeared to function well within the existing setup no one had any complaints about quality or delivery schedules and the team was largely happy with how things were. The author involved in the initiative first attempted to understand expectations for change, as a knowledge that the status quo was about to change would hopefully help both management and team to understand the need for implementing some basic processes. Management had plans for product growth and thought the team 'might grow' but didn't expect this would affect performance. During individual team interviews, the likelihood of team growth was presented and members asked to identify issues that might occur should this happen. Thirty one issues were identified: twenty six relating to growth scenario, one relating to product strategy and four relating to current issues. During team brainstorming, 'solutions' were identified and included, for example, formalisation of a team space as mitigation for cultural issues on growth, strategies for inconsistent coding style and gold-plating and the introduction of more formal version control, build, defect tracking and testing processes. Brainstorming effectively addressed contextual considerations.

A simple gap analysis with standard models simply did not help as a result of the immaturity of the organisation (they 'didn't know what they didn't know'). In order to support progression, it was necessary to establish a motivation for change (*increase in size*), support the team in identifying pending issues (*goal indicators)* and help them brainstorm ways to address these.

Both team members and management appeared 'happy' with resulting recommendations and reported plans to action these. No followup has been carried out, as yet, and so the success of the initiative is not yet certain. However, it became apparent during interviews with management that the expectation was that team would continue to contribute towards product strategic direction, a *growth* situation according to our model. Athough the approach taken supported recommendations that appeared to be appropriate for the team at that point in time, our model leads us to believe that very little will have changed and the success of the initiative will have been minimal.

6. DISCUSSION AND FUTURE WORK

The model presented in Section 4 has been created as a result of our experiences with local New Zealand software organisations. At this stage, the model has been tested only informally. We now plan to formally test some hypotheses based on the model, as discussed below.

Our first observation relates to the ’manufacturing process’ source of the popular process improvement models, such as CMMI and ISO/IEC 12207. We suggest these models are based on an assumption of stable product development whereas many small organisations are characterised by innovation and creativity. We believe the mismatch may be a contributing factor in failed SPI initiatives. Application of our model involves first identifying *growth* (i.e. business-not-as-normal) situations. We hypothesise that small organisations characterised by *growth* are less likely to achieve successful SPI outcomes because efforts must be focussed elsewhere. Our interest in this hypothesis relates to preventing doomed SPI initiatives with corresponding loss of money, time and morale.

Our second observation concerns the need to identify which goals are most important and focus improvement efforts on meeting these. Traditional models contain an implicit assumption of cost and quality related goals and the risk is that simple solutions, such as assigning a client advocate to promote client satisfaction, will be missed. The standard reference models mandate which processes are acceptable and do not support, for example, weekly meetings to clarify requirements. The 'blueprint' approach of traditional models means that, even if the traditional models were to include all kinds of activities and key indicators, they simply do not go far enough as they do not help organisations decide which gaps to close. I do not want to improve my testing process if the problem lies in clarity of requirements or if the test team is overworked and annoyed. We hypothesise that the outcomes of SPI initiatives are more likley to be favourable if recommendations are based on the identification of *goal indicators*, root causes and *context indicators*.

The key contribution of this paper is the provision of a model from which we may create and formally test hypotheses with the aim of improving our understanding of the issues surrounding SPI initiatives.

7. SUMMARY

We have suggested that a suitable model for a software system that will provide support for SPI initiatives is that of 'software system health'. The health of a human changes through time as changes to body, consciousness or environment occur. In an analogous way, the health of a *software system* changes through time as values of *key indicators* for any of *software product*, *software product owner* or *stakeholders* change. An SPI initiative may occur at any point in the *software system lifecycle* and must take into account the motivation for change, i.e. *sickness*, *growth* or *adaptation*, the *goal indicators* that inform a focus for change and the *context indicators* that must be taken into account when identifying what to change and how to change it. We have applied the model retrospectively to an SPI initiative in a small local software organisation. We have identified two hypotheses based on the model and plan to test these within local software organisations.

8. REFERENCES

Chrissis MB, Konrad M, Shrum S. *CMMI Second Edition Guidelines for Process Integration and Product Improvement.* Addison-Wesley: Massachusetts USA, 2007.

International Standards Organisation (ISO). *ISO/IEC 15504.5: Information Technology – Process Assessment – An exemplar Process Assessment Model.* The International Standards Organisation, 2006.

International Standards Organisation (ISO). *ISO/IEC 12207: Information Technology – Software Lifecycle Processes.* The International Standards Organisation, 1997.

Cater-Steel A. Process Improvement in Four Small Software Companies. *Proceedings of the Australian Software Engineering Conference (ASWEC'01)* 2001, pp. 262-272.

Grunbacher P. A software assessment process for small software enterprises. *Proceedings of the 23rd EuroMicro Conference (EUROMICRO97)* 1997, pp. 123-128.

Huack JCR, von Wagenheim CG, de Souza RH, Thirty M. Process Reference Guides – Support for Improving Software Processes in Alignment with Reference Models and Standards. *Communications in Computer and Information Science (CCIS)* 2008; 16, pp. 70-81.

McCaffery F, Pikkarainen M, Richardson I. AHAA – Agile, Hybrid Assessment Method for Automative, Safety Critical SMEs. *Proceedings of the 30th International Conference on Software Engineering (ICSE'08)* 2008, pp. 551-560.

Laporte CY, Alexandre S, O'Connor RV. A Software Engineering Lifecycle Standard for Very Small Enterprises. *Communications in Computer and Information Science (CCIS)* 2008; 16, pp. 129-141.

Chen X, Staples M, Bannerman P. Analysis of Dependencies between Specific Practices in CMMI Maturity Level 2. *Communications in Computer and Information Science (CCIS)* 2008; 16, pp. 94-105.

Aaen I. Software Process Improvement: Blueprints versus Recipes. *IEEE Software* 2003; 20(5), pp. 86-93.

Kirk D. *A Flexible Software Process Model.* PhD thesis, University of Auckland: 2007.

MacDonell S, Kirk D, McLeod L. Raining Healthy Software Systems. *The 4th International ERCIM Workshop on Software Evolution and Evolvability (Evol'08)* 2008, pp. 21-24.

Pikkarainen M, Salo O, Still J. Deploying Agile Practices in Organizations: A Case Study. *Lecture Notes in Computer Science (LNCS)* 2005; 3792, pp. 16-27.

MacCormack A, Kemerer C, Cusumano C. Trade-offs between Productivity and Quality in Selecting Software Development Practices. *IEEE Software* 2003; 20(5), pp. 86-93.

Carbonell P., Rodriguez-Escudero A.I. Relationships among team's organizational context, innovation speed, and technology uncertainty: An empirical analysis. *Journal of Engineering and Technology Management* 2009; 26, pp. 28-45.

9. AUTHOR CVS

Diana Kirk

Diana Kirk is Research Fellow at the Software Engineering Research Laboratory (SERL) at the Auckland University of Technology (AUT) in New Zealand. Diana was awarded a Masters in Computer Science from the University of Edinburgh and a Doctorate from the University of Auckland in 2007. Her industry experience spans twenty years and includes technical programming, co-directing an emerging company, project management, software quality management and consulting. Diana's main interests relate to maximising effectiveness in the processes applied during software projects.

Stephen MacDonell

Stephen MacDonell is Professor of Software Engineering and Director of the Software Engineering Research Laboratory (SERL) at the Auckland University of Technology (AUT) in New Zealand. Stephen was awarded BCom(Hons) and MCom degrees from the University of Otago and a PhD from the University of Cambridge. He undertakes research in software metrics and measurement, project planning, estimation and management, software forensics, and the application of empirical analysis methods to software engineering data sets.