Gerry Coleman – Dundalk IT

eXtreme Programming (XP) as a 'Minimum' Software Process: A grounded theory

Introduction
Within the body of literature, much emphasis has been placed on the use of process models to support software development [1, 2, 3]. Despite this, there is significant variation in the software processes used by software companies. Analysis shows that many companies are deploying proprietary, tailored "good enough" or "minimum" process in their software development activity. But *what is "minimum" process? What factors influence the composition of minimum process in software companies* and *why are these companies choosing to reject standard process models in favour of a tailored minimum?*

Research and methodology
In addressing these questions, I have focused attention on indigenous Irish software companies. I have chosen to use indigenous firms exclusively, because I can gather data, such as who established the process and how it has evolved in the organisation, which would likely not be available if I were studying a software multinational or a business in which software was a support, rather than a core, activity.

The methodology selected for use in the study was grounded theory [4]. The procedures of grounded theory are designed to develop a well-integrated set of concepts that provide a thorough theoretical explanation of social phenomena under study. The aim is to discover categories and concepts within empirically collected data, using these to generate emergent theories which are grounded in the data. Theoretical sampling ensures constant comparison of existing categories and drives the search for contrary ideas. Data collection was through structured interviews with additional documentation and artefacts used to throw further explanatory light on the studied phenomena. The software tool "Atlas TI", designed for use with grounded theory, was used to support the data analysis and category and concept generation activities. Grounded theory is especially useful, in a subject area, where there is a lack of an integrated theory which would allow credible hypotheses to be drawn, and is now becoming more widely used in software engineering [5,6].

In the study, I conducted detailed structured interviews, with chief technology officers/software development managers/project managers in 15 software product companies. The smallest company in the study had 2 software engineers whilst the largest had 120. The average number of software engineers per company interviewed was 35. The aim of the interviews was to determine what software process is in place within each company, how it varies from company to company and why.

Findings
The interview results show that, in its software development activity, each of the companies is using as little process as it feels it needs ("minimum"), and that the process itself reflects the contextual or situational factors present in each company. Using the grounded theory techniques, the coding and analysis of the interview data highlighted a range of situational factors, or variables, that act as input to a company's minimum software process. These include:

- *Background of founder/CTO/software development manager*
- *Management style/organisation culture*
- *Company size & team size*
- *Market Sector*
- *Situation Pre-process*

Whilst all of the above variables are key influencers of the minimum process a software company will use, they provide primarily a snapshot and only explain the process at a given moment in time. Consequently, there were more interesting factors to emerge from the study.

All of the companies I interviewed are software product companies. In product companies the desire to put procedures and processes in place to achieve repeatability faces the competing demand of getting the products to market as quickly as possible. This highlighted the key issue of how the engineers view process differently from management. Most of the engineers saw process as tiresome, unnecessary and something to be avoided where possible as it interferes with their 'real work' of producing code. As such, an ongoing difficulty the organisations have is to get engineer 'buy-in' to the process.

Ultimately, a successful process is one which meets the management's and the engineers' objectives. Management desire robust, quality products with extensive functionality, produced on time and to budget. On the whole, engineers wish to be creative and spend their time doing engineering tasks. They do not wish to spend their time on administration and documentation which often accompanies more process-driven organisations. As one manager said to me, "I have yet to meet a programmer who liked process."

A number of the companies concerned are turning to eXtreme Programming (XP) as a possible answer [7]. Of the 15 study companies, 3 are using XP exclusively as their development model and 7 others are experimenting with it. In the majority of these cases the engineers are driving the deployment of XP. In the 3 XP companies the model is working very well. As a manager in one said, "It [XP] is very attractive to the coders because in theory it shortens their development cycle and has them doing less stuff that they dislike doing intensely like documentation, test specs etc.. To them, it doesn't feel like a process". Management in the 3 XP companies are happy too as they are reporting faster time to market, than they had with the process they used prior to XP, with no diminution in product quality. In this way, XP becomes a model for minimum process.

Conclusions and further work

However, whilst it may help to explain why companies are opting for XP, as stated previously, the findings only represent a snapshot of the process at a given moment in time. As companies continually develop and change, the process is also required to change correspondingly, in other words to be "dynamic", thereby making a previous "minimum" process inappropriate for the company's changed environment. In this changed environment, as companies grow, they will employ more engineers and managers, expand their product range, be required to maintain legacy products and existing customers, possibly open new offices, potentially generate greater market share with more customers and possibly diversify. All of these factors will contribute to a requirement for additional process.

Therefore, the challenge for XP is, to continue to act as a minimum process, satisfy all of an organisation's business objectives and keep its engineers happy in this changed environment. How it is able to handle this is the ultimate test of the model and will determine whether it has a real future against its long-standing, more established rivals. Monitoring how XP is adapted to cope in these circumstances in the study companies will be the subject of further work.

References

[1] Royce, W.W., 'Managing the development of large software systems: concepts and techniques', Proceedings of IEEE Westcon, Los Angeles, CA, 1970.

[2] Boehm, B. W., 'A Spiral model of software development and enhancement', IEEE Computer, Vol 21. No. 5, pp. 61 – 72, 1988.

[3] Kruchten, P. *The Rational Unified Process: An Introduction (2nd Edition),* Addison Wesley, 2000.

[4] Corbin, J.M. & Strauss A., *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, Sage Publications, 1998.

[5] Seaman, C. and Basili V. 'Communication and organization in software development: An empirical study', IBM Systems Journal, Vol. 36, No. 4, 1997

[6] Carver, J. and Basili, V. 'Identifying implicit process variables to support future empirical work', Proceedings of the 17th Brazilian Symposium on Software Engineering (SBES 2003). October 2003. pp. 5-18.

[7] Beck K. *Extreme Programming Explained*: *Embrace Change*, Addison Wesley 2000.