ENHANCING EXTREME PROGRAMMING (XP) WITH ENVIRONMENT ANALYSIS

Frank Keenan[1], David Bustard[2]
[1] Department of Computing and Mathematics, Dundalk Institute of Technology,
Dundalk, Co. Louth, Ireland
frank.keenan@dkit.ie
[2] School of Information and Software Engineering, University of Ulster, Coleraine,
BT52 1SA, UK
dw.bustard@ulster.ac.uk

ABSTRACT
Software development can be treated as a form of 'problem solving'. The two main activities are then (i) understanding the problem, by analysing the situation to determine relevant requirements; and (ii) solving the problem, by implementing software that takes account of those requirements. This paper argues that eXtreme Programming (XP) currently places insufficient emphasis on problem understanding and proposes an approach that offers additional front end analysis based on Soft Systems Methodology (SSM). The models of SSM are linked to XP through user stories. The first part of the paper outlines the process involved. This is followed by details of an initial investigation of the proposal, involving experimental group projects with undergraduate computing students. The results highlight some of the strengths and weaknesses of the approach, indicating directions for future research.

KEY WORDS
Soft Systems Methodology, eXtreme Programming.

1. Introduction

Agile methods for software development are intrinsically product focused but their scope is widened by having substantial user involvement in the development process, to establish, prioritise, and verify requirements. In principle, this close collaboration, supported by feedback from the iterative delivery of software, can progressively build up a satisfactory understanding of the problem situation among the developers. The approach, however, places significant responsibility on user representatives. In XP, for example, where development begins at the requirements specification phase [1], the gap in understanding has to be bridged by the practices of *real customer involvement*, *whole team development* and the *quarterly cycle* [2].

Unfortunately, there are few guidelines on how user interaction should be implemented. In particular, it is unclear what models or system descriptions are desirable to help build a shared understanding of the problem situation and the computing solution being proposed. Various XP authors have offered advice recommending, for example, spending a great deal of time and energy in investigating contrasting stakeholder viewpoints in complex situations to help in "understanding the business" [3] and having "systematic ways for diagnosing problem situations" [4]. Agile development in general, and XP in particular, could benefit from an appropriate form of context analysis.

Soft Systems Methodology (SSM) [5] is a well-established technique for analysing the problem situation [6]. In particular, it helps identify opportunities for improvement by promoting a greater understanding by all stakeholders. It is especially helpful in complex 'messy' situations—so called 'soft' problems. It can be used in support of a goal-driven approach to change by first establishing a 'vision' of what the business could or should be and then comparing that with the current situation to identify opportunities for improvement.

Traditionally, SSM has been described as a seven-stage process [5]. In SSM, problem situations are usually captured diagrammatically as *rich pictures*. These are subjective, with no rules defined for drawing them, but they help achieve a shared understanding of a situation among stakeholders. Models of 'relevant systems' are then developed, based on that knowledge. These are expressed as *root definitions* and *conceptual models*. A root definition is a short textual statement that defines the important elements of the 'relevant system' providing a particular perspective on the system under investigation.

Conceptual models are derived from root definitions. A conceptual model identifies the activities present or implied in its matching root definition and their interrelationship. More specifically, it is a directed graph with nodes denoting activities and arcs indicating logical dependencies. These models provide a basis for further debate on the activities involved. Each activity may help identify where change is needed. Also, it might be the starting point for further analysis of the development of specific implementations. They can also form the basis of other types of model such as process models [7] and use case models [8]. Conceptual models are compared with the current real-world situation to identify 'gaps' for improvement. For each activity, the facilitator can ask: (i) is this currently conducted (ii) if so, how; and (iii) how well? Change recommendations are then derived from the results and action to improve the situation undertaken. However, a potential disadvantage of using SSM in agile development is that it is traditionally a slow process [9], designed for effectiveness rather than efficiency. Thus, some streamlining is required.

This paper suggests how SSM might be linked with XP as a pre-analysis technique The first section provides a description of how the activities and models of SSM might be linked to XP in support of the customer role. The final section describes a case study that examines the implications of the approach proposed, discussing its strengths and weaknesses and suggesting directions for future research.

2. Integrating XP and SSM

In XP, the determination of requirements starts during the *Planning Game*. The onsite customer is responsible for all 'business problems', leaving developers free to focus on technical decisions. Requirements are elicited from the customer as *user stories*. The development team then sorts and prioritises stories by value (cost-benefit) and risk. In the subsequent *Release Planning* phase, a decision is taken on which stories to include in each release and how the project should be organised overall. Each release is divided into 2-4 week iterations, each of which will complete a number of stories. During development, rescheduling may be necessary if a story cannot be developed on time. This can affect subsequent iterations, and so possibly affect the next release and the wider development plan. Table 1 summarises a proposal for how SSM and XP might be linked.

Step	Purpose
Phase 1: SSM Analysis	
1. SSM: Express Problem Situation	Build a shared understanding of the problem situation (Rich Picture)
2. SSM: Create Root definitions	Develop system perspectives of problem situation (goals) (Root Definitions)
3. SSM: Derive Conceptual Models	Produce a high level activity model for each root definition
Phase 2: Bridging	
4. Identify Possible Computing Systems	Suggest computing systems to support activities in models
Phase 3: XP Activities	
5. XP: Produce Release Plan	Compare output from 4 with current computing facilities; prioritise increments of change (Release Plan and measurable objectives)
For Release n	
6. XP: Create User Stories	
a) Analyse and develop activities	Identify activities associated with the release, expanding to lower level models as necessary
b) **For** each activity	
• define user stories	Identify and document features to support each activity (Initial User Stories)
• link models	Link each story to root definition(s) and conceptual model(s)
End For	
7. XP: Implement Iterations	
Repeat	
a) select stories	Select stories for the next iteration and prioritise them
b) perform iteration	Implement iteration, making adjustments where required
c) check project focus	Check that output of the iteration meets expectations (Revised models and plans, as necessary)
until no stories	
End Release n	

Table 1: The SSM/XP link

In Table 1 the first column identifies seven process steps that fall into three phases with the purpose and output (in brackets) of each step described in the second column. The first phase (1-3) covers an initial SSM analysis and the final phase (5-7) deals with XP activities associated with the creation of a Release Plan and User Stories, followed by the detailed definition of iterations and their implementation. In the middle is a bridging phase (4), in which possible computing systems are identified from the SSM models. The Release Plan is based on these systems, taking account of any computing support that exists already.

It is likely that senior customers, the development team, other customer representatives, and a facilitator will participate in steps 1 to 5. This is consistent with the *whole team* practice [2]. In step 6 the senior customers withdraw, leaving their interests to be covered by the *on-site customer*.

Step 6 generates user stories. As all project participants were involved in the SSM analysis up to that point this step can proceed with more confidence. First, the activities in the conceptual model that are relevant to the current release are examined. These should provide enough detail to allow the team to elicit and document desirable features. If not, activity descriptions can be taken down to a lower level of detail. The stories generated can be linked to future business value and releases. This can be implemented by adding fields to each *user story* that name the release, conceptual model activity and root definition (representing the *goal*).

In step 7, iterations are defined and performed. As each iteration is completed, the results can be evaluated with respect to the objectives of the iteration and the requirements of the customers. In effect, this ensures that the solution being developed is consistent with the problem situation under investigation. This analysis will typically influence the next iteration and may trigger adjustments to the Release Plan, the list of possible computing systems, and the higher level SSM models from which they were derived.

To be acceptable to the agile community, the benefit of using of SSM must clearly outweigh its cost. For that reason, the traditional role of SSM in identifying beneficial changes to the problem situation has been omitted from the steps described, thereby avoiding any unnecessary delay in reaching the analysis of computing needs. Also, additional analysis, such as using business use cases or rich pictures to describe existing processes has been omitted. The next section illustrates the first stages of this linked SSM-XP process in describing a small case study to assess the proposed SSM enhancement to the XP approach.

3. The Study

The study was carried out with a group of final year computing students. From a prerequisite course they had a reasonable understanding of software process, in general, including an awareness of potential benefits and challenges when following any process. Also, they were familiar with particular techniques such as requirements elicitation. They had some limited knowledge of XP, SSM and the linked approach described in this paper, based on a lecture, tutorial and directed reading.

Overall, the purpose of the study was to better understand the implications of adding an SSM pre-analysis to XP. Some of the specific questions to be clarified included: (i) how difficult is it for developers to understand SSM up to a usable level? (ii) how much additional time is needed to perform an SSM analysis? (iii) what is the benefit in maintaining the models developed? (iv) is there a need for tool support for the models used? (v) and, how would performing the SSM analysis assist the team?

At the beginning of the study, its purpose and plan were discussed with the participants, and at the end they were invited to reflect on the process followed. Exercises were conducted over five weeks during the winter semester of 2005. Students were divided randomly into two groups, of six and seven students, respectively. One group followed XP and the other the SSM-enhanced version. The groups met for two timetabled hours per week and occasionally organised additional sessions, when required. Staff could also be consulted outside the scheduled practical classes to discuss their progress and any problems encountered. For each group, a postgraduate student facilitated a number of process roles including that of the *customer* and *coach*. Each group examined the same problem which involved the timetabling facilities of a fictional university (named MADIT).

The study concentrated on user story development and incremental deployment as these were particularly relevant to the SSM-XP link. The group were, however, free to explore other aspects of XP and some attempted *pair programming* and *test-first-development*, with JUnit. Various observations were made during the study. Also, on completion, each student completed a questionnaire and was interviewed. Many of the questions required an answer in a given range 1 (strongly disagree) to 5 (strongly agree) but some were open-ended.

3.1 Illustrating the SSM-Enhanced Process

This section describes the models created in the study by the group using the SSM enhanced XP process. Its main purpose is to illustrate and further explain the process proposed.

Step1. SSM: Express Problem Situation

In the first step, a rich picture was developed to help build a shared understanding of the problem situation among the students and staff facilitating the customer and other roles. This is shown in a simplified form in Figure 1. All essential aspects of the MADIT timetabling system are identified, including an indication that students can check timetables while not on campus. As the rich picture was being developed various issues and opportunities emerged, such as, for example, a facility for lecturers to arrange preferred timetable slots in advance, and for labs to be allocated centrally by the Estates Manager based on registration figures. Other issues raised, but not addressed (represented by crossed swords) included making the facility externally available.

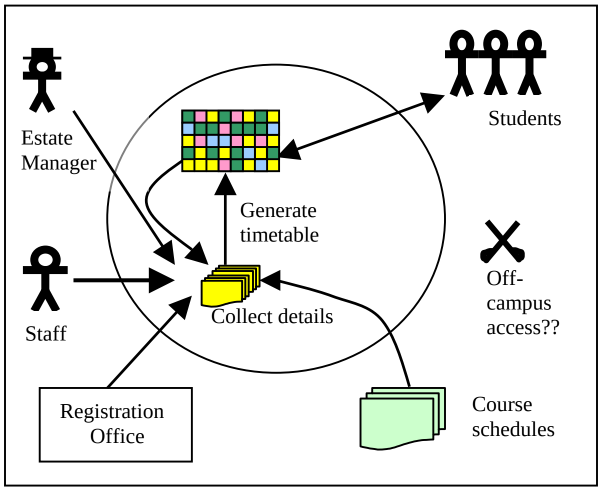

Figure 1: Rich Picture for MADIT

Step 2. SSM: Create Root Definitions

In general, an SSM analysis would examine the situation from different perspectives but here only one viewpoint was considered: *creating a reliable and effective timetabling service*. The six elements of the corresponding root definition are shown in Table 2, followed by an equivalent textual summary.

Components	**Meaning**
Customers	Staff and Students
Actors	Administrative Staff
Transformation	Prepare and produce accurate (up-to-date) timetable
Weltanschauung	Provide a reliable and effective timetabling service
Owner	MADIT
Environment	Physical resources, course schedules, enrolled students, lecturer availability

Table 2: Sample Root Definition

A MADIT owned system to provide a reliable and effective timetabling service to the staff and students, using administrative staff to prepare and produce accurate timetables, taking account of the physical resources, course schedules, enrolled students, and lecturer availability.

Step 3. SSM: Derive Conceptual Models

Figure 3 shows a conceptual model derived from the root definition given in Table 2. The model includes as the central activity the transformation taken directly from the root definition (A5). Another important activity is A12, which monitors that the defined Weltanschauung (viewpoint) is achieved, taking control action if necessary (TCA). This can affect any other activity in the model. Activities are also added to handle the environmental constraints listed in the root definition (A2, A7, A8, A10) and to cover consequential or implied activities.

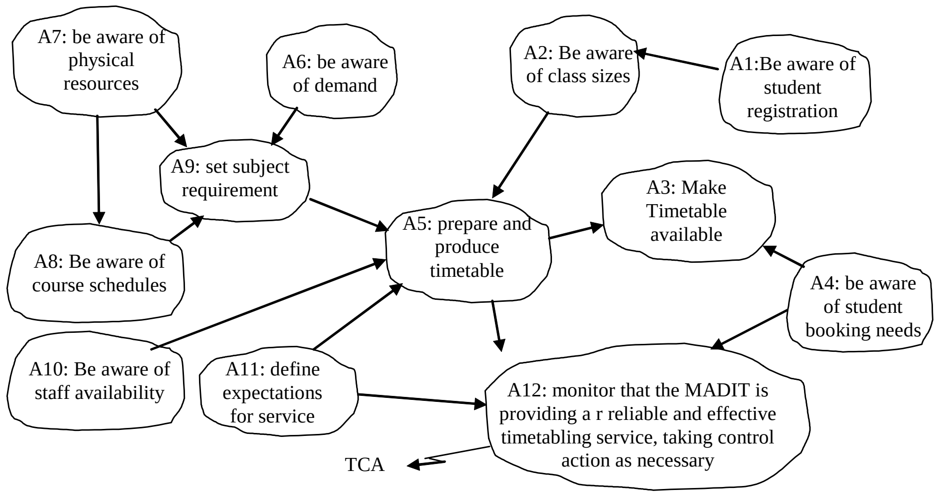
A7: be aware of physical resources
A6: be aware of demand
A2: Be aware of class sizes
A1:Be aware of student registration
A9: set subject requirement
A3: Make Timetable available
A5: prepare and produce timetable
A8: Be aware of course schedules
A4: be aware of student booking needs
A10: Be aware of staff availability
A11: define expectations for service
A12: monitor that the MADIT is providing a r reliable and effective timetabling service, taking control action as necessary
TCA

Figure 3: Sample Conceptual Model

Step 4. Identify Possible Computing Systems
Possible computing systems can be identified by working systematically through each activity in a conceptual model and noting opportunities for computing support. Each activity may suggest a new system or use a system already noted. Table 3 shows a truncated list of possible systems for activities A1, A2 and A3 for MADIT.

Activity	**Possible systems**
A1: Be aware of student registration	P1: Registration System
A2: Be aware of class sizes	P1: Registration System P2: Regulation System P3: Time-allocation System P4: Estates Management System
A3: Make Timetable available	P3: Time-allocation System

Table 3: Suggested Systems for MADIT

As a clarification of the suitability and adequacy of these systems it is useful to set them out in a diagram that shows their inter-dependency. Figure 4, for example, suggests the likely relationships among the systems identified in Table 3. The functional details of these systems are not decided until the seventh step in the development process, which deals with the implementation of XP iterations.

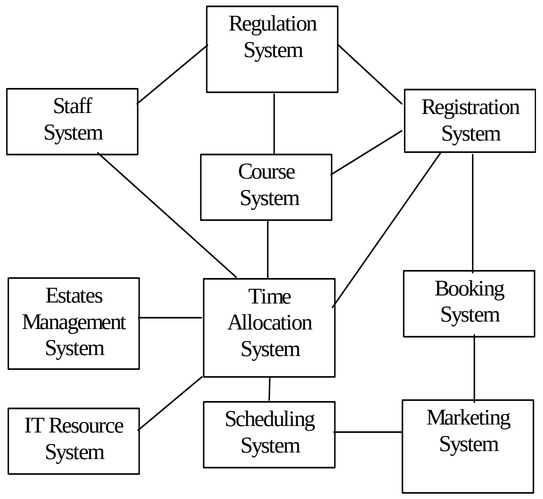

Figure 4: Computing System Dependency Model

Steps 5-7. XP: Release Plan, User Stories & Iterations
The initial motivation for the project was a request from the Students' Union for an improved service that particularly included an appointment booking facility. From the SSM analysis this has emerged as one element of a possible suite of inter-connected systems. At this stage, Senior Management at MADIT can still settle for that feature alone but will be taking the decision with knowledge of the many more forms of support that could be provided. Through discussion with the Senior Management, a Release Plan can be produced suggesting an order of implementation and the level of sophistication involved in each case. For example, a Staff System could be implemented first allowing academics to indicate preferences before a timetable is put in place. This would include preferred times to be available to meet students. This could be followed by a Booking System.

If it is assumed that the Staff System is to be implemented first then the next step is to *Create User Stories*. This involves the identification of activities associated with the release; in this case A5: *Prepare and produce timetable* and A10: *Be Aware of staff Availability*. If A10 is considered first it can be expanded to lower level conceptual models showing more detail of its activities. User stories can then be defined. Subsequently, an iteration plan is developed for this release.

3.2 Observations

In the first SSM-XP session considerable time was spent discussing the problem to be addressed and identifying the potential stakeholders. A *rich picture* was drawn on a whiteboard. It appeared that it was not difficult to draw the diagram itself but that the main challenge was to reach agreement on the problem. An initial *root definition* and *conceptual model* were produced. The diagrams developed on the whiteboard were photographed.

A new member joined the group at the beginning of Session 2 requiring more discussion and consensus-seeking, involving a reinterpretation of the rich picture and conceptual model. For the *Bridging* phase, computing systems were suggested to support the conceptual model activities. To begin phase 3, *XP Activities*, a release plan was produced. Next, the to focus shifted to release 1.

With step 6, *Create User Stories*, the high level activities of the conceptual model associated with the release were investigated. As these were exploded there was more focused discussion on meaning of each activity before further consensus was reached. User stories were agreed and estimated. Project scope was not a difficulty at this stage—a problem that did arise for the XP group. Most of Sessions 3 and 4 were spent on implementing the first release.

During these activities the team referred back to the pictures of the initial diagrams (some changed slightly) which helped to remind them of the overall objectives and to remain focused. Being able to perform activity 7c, *check project focus* allowed them to realize the progress that they had made and that this was consistent with their agreed goal and objectives. They were then able to consider the next release.

To conclude both groups presented the work up to that point. The team using the combined process had more user stories (and functionality) implemented and this appeared to be of a higher quality in terms of the detail involved. Also, although the SSM-XP group had extra effort initially in understanding SSM and creating its models, after five sessions they had reached the same stage as the group using XP only.

3.3 Feedback

This section summarises the main findings from the student questionnaires and interviews.

Generally, the process of agreeing a problem was not straightforward: "*it was difficult to reach consensus*"; "*reaching consensus did take time*"; "*diagrams were modified a number of times which meant redrawing*." Despite this, there was no evidence of wasted effort. All agreed or strongly agreed that the time spent performing the SSM analysis was indeed beneficial. Additional comments included: "*the problem was a lot clearer*"; "*good first step to achieve an understanding*"; and "*rich picture is more meaningful than 10 lines of text*". Also, defining a root definition was useful, with comments including: "*CATWOE analysis is a good guide*"; and "*it is a simple way to get overall definition*."

Most agreed or strongly agreed that performing the SSM analysis increased their understanding of the problem, increased their focus and presented an early overview of the bigger picture. Specific comments included: "*it forced communication between team members*"; "*it helped us look at the big picture and put the activities in context*"; "*it quickly got to the root of the problem*"; and "*it shows the problem in a visual way*." Generally, it was felt that the diagrams could be drawn quickly and easily. Comments included: "*SSM diagrams were not difficult to draw*"; and "*drawing them was not time-consuming.*" The response to the need for tool support to draw, record and maintain diagrams was mixed. Most felt that it would be beneficial but one person disagreed. Diagrams were consulted as development progressed. The benefits identified by the students included: "*to make sure that you are on the right track*"; and "*check objectives*." One interesting suggestion that emerged for further tailoring of the process was that the rich picture alone might be enough to set the context for XP.

4. Conclusion

This paper has argued that XP, and agile development techniques in general, can benefit from an initial pre-analysis of the problem situation to help build a shared understanding of the objectives, constraints and issues involved. Moreover, the models produced during this initial phase can facilitate XP activities, especially the creation of user stories. Details of an integrated SSM-XP process were presented and then illustrated through discussion of a student-based experimental study into the costs and benefits of the approach. The students reported favourably on the approach but, more significantly, although they took time to understand the SSM-XP analysis technique they were able to make up ground on the group performing XP alone and produced superior XP user stories. The results from this limited study have been very encouraging and more refined experiments are planned, including, for example, assessing the benefit from just adding rich picture development to XP.

Acknowledgements

The work described in this paper was supported by the Centre for Software Process Technologies (CSPT), funded by Invest NI through the Centres of Excellence Programme and the Staff Development Programme at Dundalk Institute of Technology.

References

[1] P. Abrahamsson, J. Warsta, M. Siponen, J. Ronkainen, New Directions on Agile Methods: A Comparative Analysis, *ICSE 2003*, pp. 244-254.
[2] K. Beck, C. Andres, *Extreme Programming Explained*, Second Edition, Addison Wesley, 2005.
[3] L. Thorup, O. Jepsen, Improving customer developer collaboration, *JAOO 2003*, www.bestbrains.dk/xpvip-jaoo2003-report.html [viewed 16/08/04].
[4] A. van Deursen, Customer Involvement in Extreme Programming, http://www.cwi.nl/~arie/wci2001/wci-report.pdf, pp 1-2, *XP2001*.
[5] P. Checkland, *Systems Thinking, Systems Practice* (with 30-year retrospective), John Wiley & Sons, 1999.
[6] F.A. Stowell, (ed.), *Information Systems Provision: The Contributions of SSM*, McGraw-Hill, London, 1995.
[7] D.W. Bustard, P.J. Lundy, Enhancing Soft Systems Analysis with Formal Modelling, *IEEE Requirements Engineering Symposium (RE'95)*, York, pp. 164-171, 1995.
[8] D.W. Bustard, Z. He, F.G. Wilkie, Linking Soft Systems and Use-Case Modelling Through Scenarios, *Interacting with Computers*, 13, pp. 97-110, 2000.
[9] J. Mingers, S. Taylor, The Use of Soft Systems Methodology in Practice, Journal of the Operational Research Society, 43(4), 1992, pp. 321-332.