# Supporting Distributed Extreme Programming (Extended Abstract)

Frank Maurer
University of Calgary
Department of Computer Science
Calgary, Alberta, Canada, T2N 1N4
+1 403 220 3531
maurer@cpsc.ucalgary.ca

Sebastien Martel
University of Calgary
Department of Computer Science
Calgary, Alberta, Canada, T2N 1N4
+1 403 220 3531
smartel@cpsc.ucalgary.ca

## ABSTRACT
Extreme programming (XP) is arguably improving the productivity of small, co-located software development teams. In this extended abstract, we describe an approach that overcomes the XP constraint of co-location by introducing a process-support environment (called MILOS) that helps software development teams to maintain XP practices in a distributed setting. MILOS supports project coordination, information routing, team communication, and pair programming.

## Keywords
virtual software development teams, distributed extreme programming, process support.

## 1. INTRODUCTION
Traditionally extreme programming has been limited to smaller team of 10 or so programmer. Even though lately it has been extended to larger team of up to 30 developers, sometime there is still an important obstacle; the team has to be collocated [1, 2]. It is clear that this restriction can impede development and productivity. It is not always feasible or practical to have the developer all located in the same building. We believe that distributed extreme programming will allow to reach the maximum productivity available from any programmer by providing them with the option of being part of an XP team without being physically present at the same location. There are programmers that, for some reason, are more productive developing away from an office environment. Distributed extreme programming will allow them to be more productive team members. This poster will explore the MILOS system that supports distributed extreme programming (DXP).

## 2. The MILOS Approach
The overall goal of the MILOS approach is to support process execution and organizational learning for virtual software development teams. In this paper, we focus on how MILOS supports DXP. [3] describes the knowledge management aspect in more detail.

The support provided by MILOS should be minimally intrusive to reduce overhead: MILOS stands for "**M**inimally **I**nvasive **L**ong-term **O**rganizational **S**upport". The MILOS approach can be applied for open source projects as well as for commercial teams that are distributed over the world. It was adapted to support Distributed XP. We now describe the MILOS features supporting DXP.

### 2.1 Process Support for Virtual Teams
In this section, we give an overview on the support provided by MILOS to virtual teams for process execution. More information can be found in [4]. MILOS supports the execution of globally distributed software development projects in several ways. Project managers are able to define tasks and decompose them into smaller subtasks. They can schedule them by using the Web-based user interface of MILOS. In addition, the information flow between tasks can be specified: For each task, a user is able to define expected outputs (e.g. the expected output of the "Develop workflow kernel" task is a compressed library containing a set of Java source code files that is stored in the variable "workflow kernel"). These outputs can then be used as inputs for other tasks (e.g. the variable "workflow kernel" could be the input of the task "Develop workflow user interface"). MILOS provides overviews on the current state of all tasks of a project as well as project management queries to find late tasks. MILOS also allows accessing the information produced as the output of a task easily. Team members are able to access the project plans and the information related to each task using a standard Web browser.

## 3. MILOS DXP Extensions
In this section, we briefly explain the extension to the MILOS framework to support DXP. Even though the MILOS framework nicely fits the general requirements for DXP support, we nevertheless added several extensions.

### 3.1 User Stories
After the creation of an initial project and the assignment of a project manager to it, the customer is ready to enter story cards into the MILOS system. The programmers can then contact the customer, either through the MILOS framework or by conventional means, and discuss the story with them if need be. They can revise the description of the user story – creating a new version of the existing card. In addition, they can then add notes to the story card pertaining to implementation details and split up the story into several smaller stories if the scope is too

large. They would then proceed to decompose the story into specific programming task that will be needed to satisfy the next build. The workflow engine of MILOS handles the creation and changes of tasks.

## 3.2 Task Creation

For each user story, the MILOS system automatically creates a top-level process "Design and implement user story <story number>" as part of the selected project. The input of this process is the newly created user story. Then the programmer can decompose the user story into smaller and more concrete tasks. After having decomposed user stories into concrete programming task, the programmer may describe the task in more detail using the workflow engine user interface. Tasks are associated with specific projects and can be assigned to various team members. For each task, the manager enters planned start and end dates. In addition, the users are able to define the inputs and outputs of processes. The story card automatically becomes an input of all subtasks. Furthermore, the users are able to specify the information flow between tasks by defining the output of one process to become the input of another. Specifying the information flow allows the MILOS system to provide access to input information that was created as the output of another task: the output of a task, e.g. a source code file, is transferred to the MILOS server and stored in a version management system. From there, any successor task may access the current version as well as older versions. As this is done via HTTP requests, tunneling through a firewall usually is not a problem. The programmer is able to estimate the effort and the forecasted end dates. The effort simply consists of the total number of workdays needed to complete the task (measured in ideal engineering time). To set the forecasted end date, the developer takes the required effort as well as his overall workload into account. The project manager can keep a close eye on the progress of each task by watching the "percentage complete" values and steer the team in the correct direction if the requirement for the next build will not be met. After having signed-up for some task, a programmer can pair with another programmer through the MILOS framework.

## 3.3 Pair Programming

Using Microsoft NetMeeting, MILOS provides an audio and video link between two developers and the ability to share the desktop between them. These capabilities are used to support pair programming in a distributed setting. The MILOS system keeps track of who is logged on to the system and provides the possibility to contact the responsible team member for a task or any other team member that is currently logged in. Figure 1 is a screenshot of shared desktop using NetMeeting. The screen shows the MILOS environment in the back, NetMeeting on the top right and a VisualAge for Java window at the top left. The programmer (sitting at a remote machine) just enters a method definition. The local team member inspects the code and can comment on it using audio/video conferencing. The local programmer may also take over and edit the method from his machine. The screen might seem cluttered at first. However, we tried to show as much features as possible. Normally a programmer would only look at the other desktop, see the top left corner, and switch between screens when needing to look up a function definition or when video conferencing. After the pair programming team has completed a task, they can update the task status and mark it as completed. They can also upload any output files to the MILOS server that, in turn, would route them to other team members who would need them.

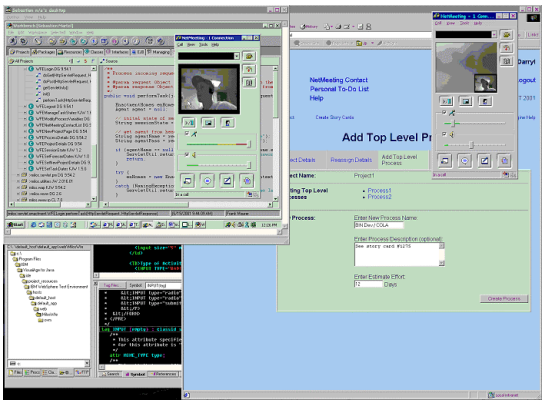


**Figure 1: Pair Programming with MILOS**

## 4. Conclusion

In this paper we have given a brief overview of the MILOS system and the tools that supports DXP, more detailed information can be found on the architecture and process support of the MILOS system in other paper [3,4]. With MILOS DXP, we are aiming at an improved efficiency of virtual teams. Whereas undoubtedly the introduction of new tools at first results in an increased workload, we argue that, in the long run, the proposed approach will improve productivity of virtual software development teams.

## 5. REFERENCES


[1] Beck, Kent. Extreme Programming Explained: Embrace Change. Reading, Massachusetts: Addison Wesley Longman, Inc., 1999.

[2] Chrysler goes to "Extremes", Distributed Computing, pp. 24-28, October 1998.

[3] Harald Holz , Arne Könnecker, Frank Maurer: *Task-Specific Knowledge Management in a Process-Centred SEE*, Proceedings of the Workshop on Learning Software Organizations LSO-2001, Springer, 2001.

[4] Maurer, F., Dellen, B, Bendeck, F., Goldmann, S., Holz, H., Kötting, B., Schaaf, M.: *Merging Project Planning and Web-Enabled Dynamic Workflow Technologies*. IEEE Internet Computing May/June 2000, pp. 65-74.