Virtual Extreme Programming Workbench: a support tool for practitioners of extreme programming in a distributed environment

Richard Nieuwenhuis
richard@mediamonks.com

Abstract

The Extreme Programming software development methodology relies heavily on the co-location of the team members. Outsourcing and teleworking is becoming more common nowadays, meaning that XP practitioners need to adapt their daily XP practices for a distributed setting. In most cases the distributed teams use (existing) tools that is a direct translation of a practice without thinking about interaction design and social interfaces. For this reason the Virtual Extreme Programming Workbench is introduced, which is a proposed tool that incorporates the extreme programming practices and adheres to the XP philosophy of being social.
1 Introduction

Extreme Programming (XP), a lightweight agile software development methodology, saw its first use in 1996 headed by Kent Beck [7]. In his book Extreme Programming Explained: Embrace Change he states a set of values, principles and practices that define Extreme Programming. Where the values and principles are too abstract to change ones behavior it is the practices that translate directly into daily activities. Some examples of XP practices are: continuous integration [12], pair programming [8], the planning game and story cards [23]. One general key characteristic of XP, or for any agile software development methodology, is the importance of a social atmosphere. Face to face communication is important and the ideal XP setting is an open office where people can see and overhear each other with ease [7][24][18]. Kent Beck even goes as far as to say that relationships between colleagues should be avoided because it would affect the social atmosphere in a negative manner.

Nowadays outsourcing and teleworking is gaining in popularity and becomes more necessary [1], so it is getting more common for a software development team to be distributed across a city, country or even on a global level. Not only does this impose cultural and time difference challenges, it also is a setting opposite of how Kent Bent envisions it to be for an XP team (not co-located). To support distributed teams that want or need to practice XP the Virtual Extreme Programming Workbench (VXPW) is proposed. The Virtual Extreme Programming Workbench is a tool that captures the XP practices and upholds to the XP philosophy of the need of a social atmosphere.

The present paper is organized as follows. Section 2 provides more detailed information on XP. Section 3 discusses distributed software development in general following with a description Distributed XP. In section 5 the Virtual Extreme Programming Workbench is introduced. Some tool support for Distributed XP is discussed in section 4 and concludes with a discussion of social software. Section 5 introduces the VXPW and its requirements.

2 Extreme Programming

Extreme Programming tries to improve software development by introducing a set of values, principles and practices. Some examples of improvements that come along when upholding to the XP philosophy are:

- improved change management.
- faster time to market.
- the quality of the software.
- customer relationship.
- improved group awareness.
- more

XP Values

Kent Beck states that values are "the root of things people like and don’t like in a situation" [7]. What he means is that individual perceptions, feelings, and historic events differ among people and directly influence the working behavior of people. One person for example might value face to face communication for important matters, but someone else values communication through e-mail. The values people have and live by directly influence the way people develop software. One person might value programming in an extensive IDE like Rational Application Developer where
the other person values using UltraEdit. Three out of five values are stated and discussed in short below:

- Communication: communication within a team is what matters most according to Kent Beck. The team as a whole has a lot more knowledge than individual persons. Developers should communicate problems with the team in early stages of the software development lifecycle, and should not be afraid to do so. Chances are that another team member already has the solution that another person is trying to solve.

- Courage: how one acts with fear. If you know a fairly risky solution for a problem, the way you act or don't act can be critical.

- Respect: the whole team should respect each other.

**XP Principles**

Principles are domain-specific guidelines for life [7]. The deeper meaning of this is that values are too abstract to guide behavior, and in the case of XP, software development behavior. The intention of the XP principles is to ease the search for XP practices that fit in a (mostly) unique working environment. Some principles are:

- Humanity: XP states that the (social, working) atmosphere has direct influence on the development skills of a developer. When developers are not comfortable with their working environment, it will influence the quality of the software he/she produces which also influences the overall team quality. Some examples of what people need to be good at or uphold to in software development are:
  
  freedom from hunger, physical harm, fear of job loss.
  the ability to improve skills and perspective.
  to understand and to be understood by others of the team.

- Economics: people's awareness that what they are doing has business value, meets business goals, and serves business needs.

- Reflection: it is important to reflect from time to time. This can be an individual reflection of yourself or a reflection on the entire team. Information from reflection moments can come from different sources like analyzing the written code, by asking other team members, the customer etc.

**XP Practices**

The practices of XP are the daily activities of the whole software development team. Not all practices have to be followed to do XP so only those practices are chosen by a software development team that fit a particular situation. The practices are divided into two categories, namely: primary practices and corollary practices. Some of the primary and corollary practices are stated next.

- Primary practices are practices that one can start doing with minimum difficulty.

  Sit together: the whole team should work together in an open environment. Individual privacy needs of team members can be accomplished by having small private spaces nearby or limiting the work hours.
User Stories: create small stories so a planning can be made with customer readable functionality.

Pair programming: develop with two people behind one machine. Create a working environment where two developers can sit comfortably side by side. Pair programmers keep each other on task and clarify ideas through direct face to face communication. With pair programming one developer is typing in the code called the *driver*, and the other developer reviews the code typed in by the driver. The role of this person is called the *navigator*. Every hour or so the roles between the pair should be switched.

- **Corollary practices** are practices that can be difficult to start doing without having proper experience with the primary practices.
  - Real customer involvement: make the customer part of the team.
  - Root-Cause analysis: solve a defect and find its cause and keep the solution for everyone to find. In this way the team will learn from it and hopefully never make the same mistake again.

For all details on XP see [7].

### 3 Software development in a distributed environment

As mentioned before, most (agile) software development methodologies state that development teams should be co-located in an open environment. Some reasons why the co-location of team members is beneficial for software development are summarized below:

- face to face communication.
- daily meetings are arranged with ease.
- improved project coördination.
- people can see who is doing what "instantly" which improves group awareness.
- more efficient knowledge management. An example is the often cited coffee break meeting where knowledge about the project is being shared without even realizing it.
- different people have different knowledge and people know whom to get information from. People can just walk to a specific person with ease.

It is obvious that many of the above stated reasons are directly related to communication. Communication is one of the most important aspects of software development and it is this aspect of software development that is challenging to do efficiently in a distributed software development environment [14] [19]. A lot of development teams are struggling to find practices that are suitable for a distributed situation of which the following is an example. Figure 1 shows a very inefficient asynchronous communication line through e-mail based on a real situation witnessed at a big software company in the Netherlands. The figure shows the route of a "small" change request that was send to a product manager of the software company. The product manager forwarded the request to the Dutch development team who in turn found out that this request had affect on a part that of the system that was developed in Malaysia. The head of the development team forwarded the request to a development team in Malaysia. This team in Malaysia eventually forwarded it to another team in Malaysia and finally after four days an answer was received about the change request.

4
Only recently the software world is recognizing the importance of good adaptations of (agile) software development methodologies for distributed situations. An increase in research of distributed software development is witnessed as of 2006 [14]. Software development where a team is not co-located is called Distributed Software Development (DSD) and if the distribution is beyond national borders we speak of Global Software Development (GSD) [10]. Research on GSD is diverse and there is still no well established distributed (agile) development methodology as is the case with co-located methodologies. Different literature on GSD propose or follow different distributed practices where two of them are discussed in short next.

The patterns & practices group at Microsoft have been practicing an agile development methodology in a distributed setting for more than five years. In [19] followed practices by the group are described where some of them are cited below:

- Tactics for communication: each team has a dedicated conference room and projector which is always available for setting up quick (distributed) meetings. Every team member has a headset and desk phone that can be used for quick communication with team members situated elsewhere or for remote pairing.

- Travel for some face-to-face communication: site visits are done by remote developers at the start of a project and for one to two every six weeks during the project. In this way the different team members can put a face on the ‘digital’ persons they are interacting with during the project.

- Tooling: the group try to use distributed tools that do not disrupt the natural flow of work, meaning that overhead in work of using the different tools should be minimized. Examples
of tools that are used by the patterns & practices group are Visual Studio Team System and Scrum for Team System.

In [21] nine principles are described that originate out of ten years of experience with virtual teamwork. The article does not focus primarily on GSD but on distributed teamwork in general. However, many of the challenges distributed software teams face are also faced by distributed teams in other lines of work. Some principles discussed by the article are summarized below:

- Plan (virtual) activities that cause people to get to know each other. Leaders of a virtual team should design activities that increase teambuilding. Face-to-face meetings from time to time are also important.

- Reward structures must be realigned for virtual teams. Virtual team members often miss out on the appreciation and face time common for co-located teams. In a distributed setting an early presence or a good achievement might not be noticed by the senior staff and might be less valuable in performance reviews.

- Standards and terminology should be agreed upon. Different virtual team members may use the same terminology for different meanings. With team members that are co-located this is less of a problem since face-to-face communication and body language show valuable clues of what someone is talking about. Virtual teams often use e-mail, an instant messaging system, or a wiki for communication that would otherwise be done face-to-face. It is easy to see that this can cause more miscommunication than face-to-face communication. A real live example that caused a lot of problems eventually was the use of the term **White label** in a project that involved an educational website. The website supported many white labels (unique environments for different customers) and by mail a request was made by the client to add another white label. Once the white label was finished the client noted that the new white label was not as envisioned. Further questioning revealed that a new white label was not the actual request but that was a general feature request for the website. One week in total was lost due to this miscommunication.

- Integrate collaboration technology into daily work.

Although different practices and principles for GSD are suggested and used, a general pattern can be observed from the different situations. Suitable practices and principles are created to replace the communication barriers and tooling is used to replace common co-located software development practices.

### 3.1 Distributed Extreme Programming

Distributed Extreme Programming (DXP) is an adaptation of Extreme Programming enabling practicing XP in a distributed environment. XP in a distributed setting is clearly not how Kent Beck envisions XP being practiced since he only mentions it briefly in his book. Also, throughout his book he points out on multiple occasions the importance of a social atmosphere and the co-location of team members. But as XP is quite popular as a development methodology and the distribution of team members is common practice nowadays Distributed Extreme Programming becomes a necessity for those who practice XP. Research on DXP is still relatively scarce but is steadily gaining momentum.

In [16] DXP is introduced. The paper starts with dividing XP practices into two groups, namely: practices that require co-location and those that do not. It are the practices that require co-location
that need to be adapted in some way in order for the practice to work in a distributed environment. The table below provides some examples of practices that do or do not require co-location.

<table>
<thead>
<tr>
<th>Practice</th>
<th>Requires Co-location</th>
<th>No Co-location Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>the planning game</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pair programming</td>
<td></td>
<td></td>
</tr>
<tr>
<td>continuous integration</td>
<td></td>
<td></td>
</tr>
<tr>
<td>on-Site customer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>collective ownership</td>
<td></td>
<td></td>
</tr>
<tr>
<td>coding standards</td>
<td></td>
<td></td>
</tr>
<tr>
<td>small releases</td>
<td></td>
<td></td>
</tr>
<tr>
<td>testing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The grouping depicted in the table above however should not be as strict as [16] claims it to be. Some of the practices placed in the "no co-location required" group do impose some minor additional requirements in GSD. For example the practice collective ownership generally uses a local Subversion server when the team is co-located. In a GSD setting the Subversion server should be reachable by the distributed team imposing more security challenges. The other way around is also possible. The strict co-located practice continuous integration could be done in distributed environments with the use of the tool CruiseControl [4] for example.

XP heavily relies on communication [17] between team members, which is evident from its values 2, the planning game practice together with the user stories and the wall [23]. For this reason assumptions are made in [16] of the existence of several conditions and tools. For example, familiarity between team members should exist. DXP is expected only to succeed if the team members know each other on a personal level as is also discussed in [19]. Another assumption is the availability of video (and audio) conferencing software for effective communication. Software called Web-Desktop was the main goal of [16] which is a central working environment for DXP.

[22] also discusses DXP and introduces a tool named TUKAN. TUKAN is a synchronous distributed pair programming environment that incorporates some results from the groupware domain [11].

4 Tool support in DXP

Almost all literature that discusses GSD or DXP do also discuss one or more distributed software tools. This makes it obvious that, in order to practice DXP, the use of the right tools is an essential part of it. This section discusses some existing tools that distribute one or more XP practice(s).

TUKAN

Besides a remote pair programming environment, TUKAN [22] also provides the activity explorer that can be used for the planning game. TUKAN also incorporates chat capabilities between team members and can also instantiate audio conversations if bandwidth allows it. Figure 2 shows a screenshot of the pair programming environment of TUKAN. To identify different user roles in a TUKAN session different cursors are used and the distributed team members are visible through the use of avatars.

Quite popular is tool support for the pair programming practice where two developers can each sit behind their own machine. Especially in the last couple of years several tools have been introduced that enable remote pair programming with two developers sitting at their own (separated) machine.

Cola

Cola [2] is a real time shared editing plug-in for the Eclipse IDE. It enables software developers to edit the same code remotely at the same time. Changes made by one developer are immediately visible to the other developer. Cola completely ignores the driver and navigator support and does
not include direct communication through chat or voice over IP meaning that Cola users need to rely on third party tools which is evident in the screen cast at [3]. The single programmers role means that software developers in a Cola session can edit the same code at the same simultaneously. It is of course possible that agreements are made between developers on the two different roles but it is just too easy to change a small part of the code as a navigator which in effect can annoy the driver.

Sangam & Ripple

Sangam [13] is another distributed pair programming tool but also lacks driver, navigator and communication support. The shared editing architecture has proven to be sufficient which is why Ripple [9] extends on Sangam. Ripple introduces chat capabilities between pair programmers where preliminary results are positive.

The main problem of the briefly discussed tools and many others, besides mostly only focusing on a smaller part of XP, is the use of many smaller tools together or a single tool that lacks in the main characteristic of XP, being social. Besides the obvious of enabling communication through chat, voice over IP together with video there are also other enhancements possible to increase the positive social feeling a person has with software. Software having a social atmosphere is discussed next.

4.1 Social software

As mentioned, a co-located XP team improves the social atmosphere because it simplifies: inter team member communication, arranging team meetings, pair programming, team awareness etc. Most DXP tools translate a XP practice directly into a digital form without taking much thought
on the social part of XP and proper (interaction) design. An example we witnessed on several occasions is the use of the tool Trac [5] as a digital version of the XP wall. Not only does Trac lack in design to be a digital replacement of the wall, team awareness is also lower.

Social software is software that enables group communication and computer-mediated communication [6] which is essential for DXP. There is however also evidence that incorporating social cues into the Graphical User Interface (GUI) will have a positive effect on the period of time a user will sustain his/her social state with the tool [15]. Personalized messages and the use of avatars when interacting with team members in other ways than video conferencing are two examples of social cues. Another factor that can contribute to a more positive social feeling one has with software is the use of humor in the GUI [20]. An example are the error messages displayed by the browser Google Chrome. Instead of the "normal" not meaningful error codes Chrome displays a "fun" text message with a smiley. An example of a Chrome error message is shown in figure 3.

![Google Chrome error message](image)

Figure 3: Google Chrome error message

Different social cues, humor and proper design can improve the social feeling a person has with software which means the look & feel (graphical design) of a DXP tool, besides the functional implementation, is a crucial factor to its success. It is often the for mentioned social elements existing DXP tools lack in. As an example figure 4 shows the Ripple interface.

As can be seen from figure 4 the shared editing window is just a plain Eclipse editor with a basic chat window with almost no social cues. Also, a team member with no technical background will have a hard time understanding these types of GUI’s.

A simple example of how much a design can have a positive contribution to the overall feeling with the software is shown in figure 5 and 6. Translating story cards and the XP wall directly into a digital form is one of the XP practice that should not be digitized directly [18]. In an XP software development team the XP wall immediately shows the status of an iteration(s) and improves team awareness (who is working on which requirement). The wall is also understandable for everyone in the team and not just the technical persons. Figure 5 shows a hypothetical digital XP wall. The different story cards are visible and are placed in their respective iteration. The status of a story card is presented by having a different color. Place holders with names are visible for story cards that team members have taken to work on. This digital wall could also be accessed in the Eclipse IDE. In Eclipse the different story cards can be opened by clicking on them which is visible in figure 6. After having clicked on a story card more card information is presented and avatars are shown of the people working on them instead of only their names. To give the digital wall and the different cards a "team feeling" the background and colors could be changed also further improving the social feeling of the digital wall. It would even be possible that the different groups of a distributed XP team put the digital wall on a big screen at their respective locations which has proven effective [18]. To add even more possibilities, if a touch screen is used the story cards could even be clicked on and opened just like in Eclipse.
In the next section a new tool, the Virtual Extreme Programming Workbench, which enables practicing DXP.

5 Virtual Extreme Programming Workbench

The Virtual Extreme Programming Workbench (VXPW) is a new tool that (eventually) integrates all XP practices for a distributed setting and that upholds to Beck’s XP must have, namely: the need of a social environment. The different research on DXP en existing DXP tools leads to several requirements the VXPW must uphold to. The different requirements and assumptions are stated and discussed next:

- Recognizable software environment for XP practitioners. Simply directly digitizing all the XP practices is not sufficient. The functional flow of the different digitized practices in the VXPW should feel natural and easy to use for XP practitioners.
- The workbench should feel social for its users. An integrated communication environment should be available. DXP team members must be able to communicate with ease through chat, audio and video. Besides the communication possibilities the look and feel is important (see section 4.1). Social cues must be used wherever it contributes to the social feeling of the VXPW. For this reason, through research, a set of design guidelines must be set up. The design guidelines will be gathered by having a control design and a social design that is tweaked in several iterations. After every iteration the result of the social graphical design will be measured by conducting a validation study. In the different validation studies the
participants will be presented by one of the two designs randomly. By answering specifically
designed questions a measurement of the social feeling of the design will be possible to
calculate, eventually arriving at a proper set of design guidelines [15]. The social environment
of co-located XP teams leads to proper team awareness meaning the VXPW should also be
high on team awareness.

- Reusing existing tools for different parts of the workbench should not compromise the created
look & feel and functional flow of the VXPW. If for example Skype could be used for voice
over IP communication it should be integrated in such a way that users do not have additional
overhead and restrictions of using "another tool inside a tool". Opening the help functionality
in the (Skype) communication environment should not open the Skype help, but the VXPW
help functionality.

- Asynchronous usage possibilities. With DXP it is possible that different team members are
located in different time zones imposing inter team collaboration challenges. Other than
possibly making agreements on work times the VXPW needs to incorporate asynchronous
functionalities that will lower the difficulties that arise with time zone differences.

- The system should be accessible by everyone in the team. Team members with no technical
background should be able to access certain parts without the need of the Eclipse IDE for
example. A part of the VXPW will most likely be available and accessible through an internet
application. Web pages are usually more user friendly than a full fledged IDE like Eclipse or
Netbeans.
The first focus of the VXPW will be the user story practice together with the wall. This part of XP contains all parts of XP that are relevant for digitizing XP and is thus a good starting point of the VXPW.

6 Future usage

As mentioned the VXPW will eventually incorporate XP as a whole. This section describes a hypothetical use case of future VXPW usage between XP team members.

Two DXP team members have agreed to pair program on a user story on a certain time. The pair programming appointment could be made through the VXPW by using the chat or voice over IP possibilities and the agenda or just by phone. The first team member that starts working picks the chosen user story either by using the digital XP wall (see section 4.1) by using a computer or a potential touch screen available in the working environment of either team member. This process is visible in figure 7.

Both team members log into the VXPW making themselves visible to each other through a user list (similar to chat tools like MSN messenger or Yahoo messenger). One team member initiates the pair programming session by selecting his pair programming colleague. The VXPW asks for confirmation to the other team member and after he/she confirms the pair programming environment is initialized. Both members can agree upon driver and navigator roles and the session can start. The team members can also initialize chat or voice over IP sessions. While programming
both team members encounter a strange problem they cannot solve but know that another XP team member has had a similar problem. They send a message through the VXPW to that person and ask him if he wants to take a look. He does not mind to take a look and requests to join the pair programming session. The pair programmers accept and when the third person has joined the session one of the pair programmers (if there is a driver role it will be the driver) shows the code that gives the problem. Once the session is finished one of the team members can put the story card back on the digital wall.

The case above is just one of the daily DXP routines that the VXPW will be offering in the near coming future.

References


