Build Notifications in Agile Environments

Abstract. In an agile software development environment, developers write code that should work together to fulfill the wishes of the customer. Continuous integration (CI) ensures that code from different individuals integrates properly. CI compiles the entire codebase, deploys and tests it with each change. CI alerts developers of any problems as errors can be fixed more easily if caught earlier in the process. This paper compares the effectiveness of different types of mechanisms for notifying developers to a successful or unsuccessful build. Two different quantitative and qualitative user studies were performed testing the effectiveness of three types of notification devices – one virtual e-mail based mechanism, one using ambient lava lamps, and one robotic device. The results show that most developers preferred an easily visible but unobtrusive ambient device combined with an e-mail describing the problem in more detail.

Keywords: Agile methods, information awareness, ambient display, continuous integration

1 Introduction

Agile methods [1] are becoming popular in the software industry. In agile software development projects, there may be anywhere from one to hundreds of developers working concurrently on the code base, so it is imperative that all software written by each developer integrates properly into the entire project. To this end, most agile teams adopt *Continuous Integration* (CI). CI is the practice of automatically compiling, deploying and testing the entire codebase against a suite of prewritten tests. This occurs after any change to the codebase, usually multiple times per day.

When integration is finished, it is important for the developers to become aware of the result so that any problems can be immediately fixed. If a problem goes undetected, other developers may synchronize with a broken version of the codebase, and this may result in increased effort required to fix the problem and delays in integrating their changes to the latest build. Thus, awareness of the build status is essential, especially after submitting new code to the codebase.

In this paper, we evaluate notification mechanisms of three principal categories: virtual, ambient, and active. Each has its own advantages and drawbacks. The goal of this research was to compare different notification mechanisms within the context of an agile software development environment to determine which is more effective in ensuring that build problems are fixed as soon as possible.

2 Previous Work

In agile teams, it is important that all developers work with the very latest working version of a codebase. Every change in the code, however minor, should be integrated continually into the codebase, instead of postponing code integration to the very end as a development cycle. Several continuous integration tools have been developed. Examples of such tools include Apache Ant/Anthill [2] and CruiseControl [3]. These tools provide logging, compilation, and executing automated unit and acceptance tests. They can email build results to user specified recipients, or can execute user specified programs on a successful or failed build.

A study by Saff and Ernst [4] evaluated continuous integration when used by a single developer to ensure new code passed regression and unit tests. They found that continuous integration had a positive effect on the completion of programming tasks. Their study shows that even individual developers benefit from continuous testing of their own code. Our research focuses on CI tools that affect agile teams as a whole, specifically, how they can be notified when build breakages occur.

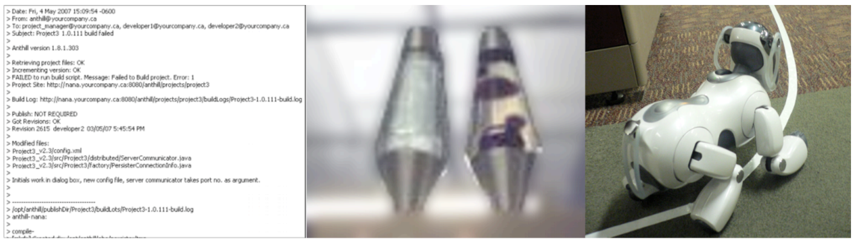
Cadiz et al. [5] introduced three different classifications in which information awareness techniques generally fall. *Polling* requires users to access information themselves to check for an update. This involves a high cognitive burden as users need to actively remember and act to access the required information. As a result, updates may be missed. *Alerts* bring information to the user, demanding attention. The chance of missing a potentially important update is reduced, but alerts can become a distraction. There have been studies exploring the effects of distraction and interruption showing the disruptiveness of instant messaging alerts. McFarlane compared different approaches to user interruption and found that generally, people "when people are forced to handle interruptions immediately, they get the interruption tasks done promptly but they make more mistakes and are less effective overall" [6]. Cuttrell et al demonstrated "the harmful effects of notification delivery on memory for the prior task early in a task's lifecycle" [7]. Rather than take focus away from the task at hand, *Peripheral Awareness* takes advantage of our subconscious and our ability to absorb information from the environment without having to consciously concentrate on its delivery.

Alberto Savoia [8] created an ambient build notification system using peripheral awareness to inform developers of the build state. The system uses two lava lamps, one red and one green. The lamps are connected to a wireless transceiver that controls power flow to them. If the build is successful, power is supplied to the green lamp and denied to the red lamp. If the build fails, the red lamp is turned on and the green turned off. The transceiver emits an audible *snap* when activated, adding to the development soundscape [9]. The presence of bubbles in the lava lamps indicates a sense of time, as the lava only appears when the lamp has been on for more than about ten minutes. However, there have been no empirical studies showing the effect that lamps have on the development process.

3 Experimental Setup

3.1 The three modes

The goal of the experiment described herein is to evaluate three notification mechanisms (virtual, ambient and active – Figure 1) within the context of a shared project, where notifications are sent out when code is committed. The goal was to determine which of the three modes would be most effective for an agile team.



```
> Date: Fri, 4 May 2007 15:09:54 -0600
> From: anthill@yourcompany.ca
> To: project_manager@yourcompany.ca, developer1@yourcompany.ca, developer2@yourcompany.ca
> Subject: Project3 1.0.111 build failed
>
> Anthill version 1.8.1.303
>
> Retrieving project files: OK
> Incrementing version: OK
> FAILED to run build script. Message: Failed to Build project. Error: 1
> Project Site: http://nana.yourcompany.ca:8080/anthill/projects/project3
>
> Build Log: http://nana.yourcompany.ca:8080/anthill/projects/project3/buildLogs/Project3-1.0.111-build.log
>
> Publish: NOT REQUIRED
> Got Revisions: OK
> Revision 2615  developer2  03/05/07 5:45:54 PM
>
> Modified files:
> Project3_v2.3/config.xml
> Project3_v2.3/src/Project3/distributed/ServerCommunicator.java
> Project3_v2.3/src/Project3/factory/PersisterConnectionInfo.java
>
> Initials work in dialog box, new config file, server communicator takes port no. as argument.
>
>
> ------------------------------------
> /opt/anthill/publishDir/Project3/buildLots/Project3-1.0.111-build.log
> anthill-nana:
>
> compile-
```

Fig. 1. The three modes: email, *Java Lava Lamps*, and BuildBot.

There were several ways in which participants could access build information for the project – polling (checking the build status webpage), active alerts (BuildBot), peripheral awareness (lava lamps), and via virtual means (e-mail). Our evaluation compares the different notification mechanisms within the context of an example project. Notifications were sent out automatically by the continuous integration server whether the build resulted in a failure or success. After the study finished, participants were interviewed and answered a questionnaire after the study and they expressed their feelings about each of the above mechanisms.

A build status webpage is a display of detailed build information. Anyone interested in the build status history can visit the page to check for updates. The information is quite detailed, however, updates may be missed since the user must consistently check the page (and reload it in the browser) in order to receive the updated information.

CI tools have an email option to send messages to developers when integration occurs. Email can be seen as both a polling or an alert mechanism, depending on the e-mail client used by the recipient. An email program can wait for the user to check for new mail (poll), or it can present a pop-up to alert the user of newly arrived mail (alert). Email is (usually) delivered in a very short amount of time and can be very detailed, however, too many emails can result in an information overload and the recipient might stop reading them altogether.

Java Lava Lamps were used as an ambient device in this study. An ambient display is a way of keeping people informed about the build state without causing a disruption. However, they must be noticeable enough so that developers can easily access the information. An ambient device whose information is not accessible must be polled and thus the advantage of ambience is lost.

The BuildBot [10] robotic notification device was designed as an active, ambient build notification tool to study the effect of such a device on an agile team in the context of a shared project. The robot is an active ambient build notification mechanism - using motion, sound and presence. If the build fails, BuildBot follows a network of lines to reach the responsible developer's workstation and kindly barks until the build is fixed. BuildBot is ambient and adds an element of fun to the development environment. However, it runs the risk of becoming too loud or intrusive.

3.3 Experiment Setup

A case study was performed in which the three modes were tested in an academic agile environment. The environment was a laboratory with six-foot high opaque cubicle walls. When the lava lamps were being tested, they were placed on top of a high cabinet in the centre.

Participants included 8 graduate students working full-time and part-time on a common development project, which had been in development for 48 months. There were also 4 *observers* – participants not involved with the project but co-located within the development environment. Developer participants were able to work as they would normally on the project – including programming in pairs, listening to music, or working at home. This was done to create an environment that resembles the developers' natural environment as closely as possible.

During the first week, email was sent only to the developer responsible for a build breakage. During the second week, a pair of *Java Lava Lamps* were installed in a central location and showed the build status. The lava lamps were removed at the start of the third week and BuildBot was used as the physical notification device for the third week. The build status page was available to all participants throughout the entire length of the study, and email was sent to developers who committed code.

There were no build breakages during the first week, but emails indicating a successful build were sent to the developer who had last checked in code. One build breakage was logged during the second week, and one was logged during the third.

4 Results and Discussion

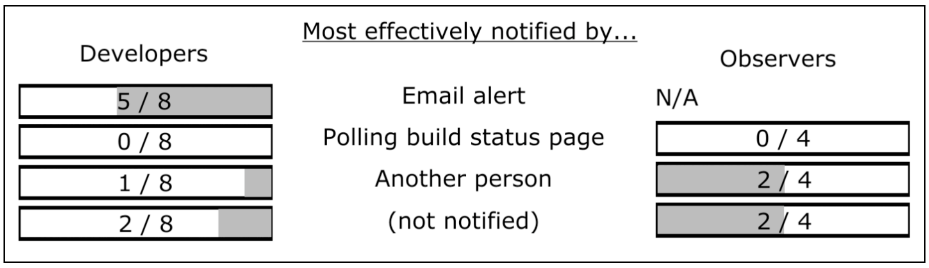
3.1 Pilot Study

A pilot study was performed previously with an agile development team under similar conditions. The purpose of this study was to investigate developers' impressions of each of the three modes described in Section 3.1. As developers worked on unrelated projects, notifications were sent out at random and they would indicate via instant messenger when they had noticed the notification. Thus, these notifications had no meaning to the developers, since they were sent out at random rather than in response to the result of a build process.

In this paper, we compare BuildBot against email alone (no physical notification) and Savoia's lava lamps. Study participants worked on unrelated projects, and the notifications were sent out at random. Response time was measured by time an instant message was sent to the experiment facilitator. The results were that few people noticed a change in the lava lamps, and participants indicated they were more effectively alerted by the robot. The average time for noticing the robot was 3 minutes - and that they also found it distracting.

4.1 Experiment results

During the first week, participants who used email as an alert (by employing an email notification program) felt they were aware of the build more than those who did not. Even though the build page was available and easily accessible, not a single participant polled it. Five of the eight developer participants indicated they were notified more effectively by email than by polling the build status page. Two observers overheard conversation about the build status. Two of the developers did not receive emails, did not check the build status page and were not alerted by other people about the state of the build. They were totally unaware of the build status.

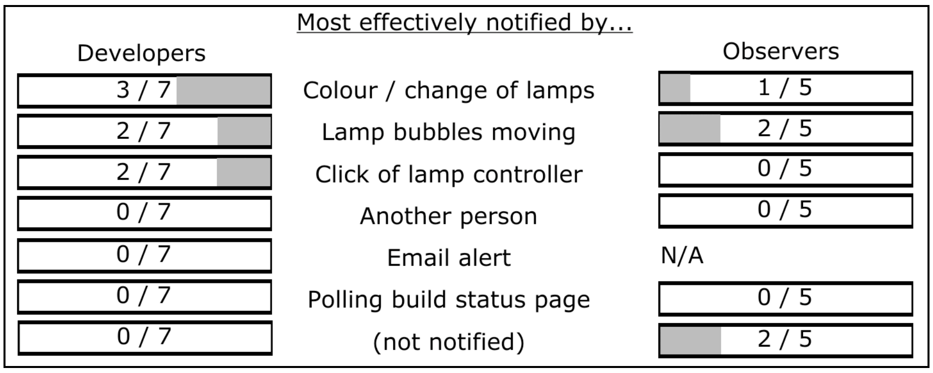


Developers	Most effectively notified by...	Observers
5 / 8	Email alert	N/A
0 / 8	Polling build status page	0 / 4
1 / 8	Another person	2 / 4
2 / 8	(not notified)	2 / 4

Fig. 2. Results from Week I.

For the second week, two lava lamps, one red and one green, were placed in a prominent location (on top of a cubicle shelf) in the centre of the laboratory. These lamps' power transceiver received signals from the continuous integration server, lighting either the red or green lamp, depending on the build status. The continuous integration suite also sent an email to the developer who had committed the latest code.

Of the seven developers (one of whom was unavailable), three were better notified by email and four by either the colour change or movement of the lava lamps. While some participants did check the build status page after the noticed a change in build status and some were alerted by other people, all participants were alerted more effectively by either the lava lamps or by email. These results are shown in Figure 3 below.



Most effectively notified by...

Developers		Observers
3 / 7	Colour / change of lamps	1 / 5
2 / 7	Lamp bubbles moving	2 / 5
2 / 7	Click of lamp controller	0 / 5
0 / 7	Another person	0 / 5
0 / 7	Email alert	N/A
0 / 7	Polling build status page	0 / 5
0 / 7	(not notified)	2 / 5

Fig. 3. Results from Week II – developers and observers.

Participants – both developers and observers – felt that the lava lamps were a fun addition to the development environment and were not distracting. However, some participants felt that they could not properly see the lamps and thus this form of build status information was unavailable to them at their desks, and could only see them when entering or exiting the room. This may be the cause of them being generally less aware of the build (Figure 4).

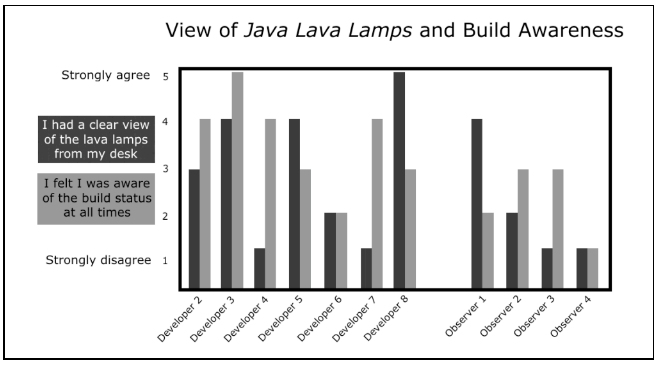


Fig. 4. Results from Week II – view of lamps from workstation and build awareness.

For the third week of this study, BuildBot was set up, and the lava lamps were removed from the development area. Only two out of the twelve participants felt that they were aware of the build at all times (Figure 5). This may be due to the fact that BuildBot emits no indication of a successful build. This was different from a lava lamp, whose glow is a constant reminder to the developers and thus accessible to those with a clear view.

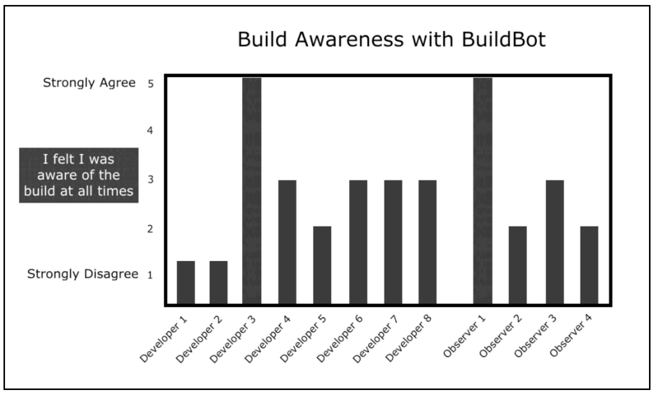


Fig. 5. Results from Week III – Build awareness with BuildBot.

After a build failure, BuildBot was generally more noticeable than the lava lamps, but became more distracting for the developers (Figure 6). Some developers liked the fun nature of the robot and welcomed the distraction, while others vastly preferred a quiet environment conducive to productivity, with no such distractions. One of the developers noted that "*initially it would be the novelty effect, and over time it would just become annoying, more than anything*."

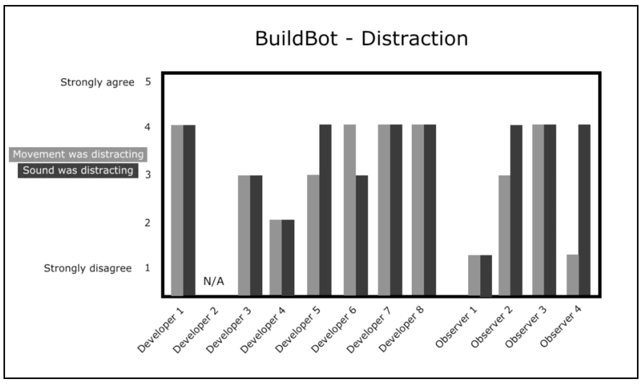
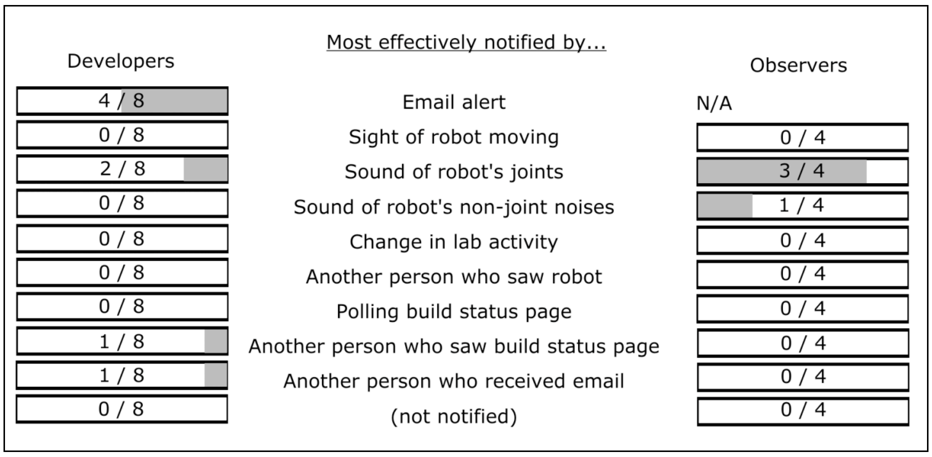


Fig. 6. BuildBot - Distraction.

After the participants had seen all three notification devices in action, they were asked to compare them. Developers indicated they were more effectively notified by email, and observers indicated they were more effectively notified by the sound of the robot moving (Figure 7).

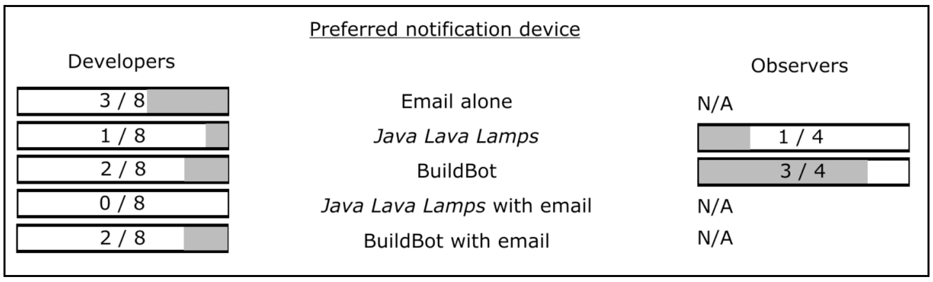


Most effectively notified by...

Developers		Observers
4 / 8	Email alert	N/A
0 / 8	Sight of robot moving	0 / 4
2 / 8	Sound of robot's joints	3 / 4
0 / 8	Sound of robot's non-joint noises	1 / 4
0 / 8	Change in lab activity	0 / 4
0 / 8	Another person who saw robot	0 / 4
0 / 8	Polling build status page	0 / 4
1 / 8	Another person who saw build status page	0 / 4
1 / 8	Another person who received email	0 / 4
0 / 8	(not notified)	0 / 4

Fig. 7. Comparing notification mechanisms in Week III.

Participants were then asked which notification device they preferred. Developers had different opinions, but seemed to prefer having email as part of the notification. Observers liked the robot, possibly because they were seated farther away from the robot's path and thus were less distracted, or because of the novelty effect (Figure 8).



Preferred notification device

Developers		Observers
3 / 8	Email alone	N/A
1 / 8	*Java Lava Lamps*	1 / 4
2 / 8	BuildBot	3 / 4
0 / 8	*Java Lava Lamps* with email	N/A
2 / 8	BuildBot with email	N/A

Fig. 8. Participant preferences.

4.2 Discussion

The build status webpage was not used very much at all, even though it was available throughout the entire length of the study. As one developer said, "*going to the website is not my idea of a notification system. It defeats the purpose*."

Email was preferred by some developers because each message is nearly instantaneous, simple, not location dependent, not obtrusive to others, and shows the entire build break message such as tests failed. Some developers noted that important email messages could be missed if the developer is not using a popup notification program, or that too many email messages can become a form of spam and thus become ignored.

Some participants liked the lava lamps because they were simple, unobtrusive, and added a bit of fun to the development environment. However, most participants did not notice the lamps because of the cubicle walls. For those out of sight range of the lamps, they effectively became a polling device. Also, since the lamps only show red or green, some participants felt that the information was too limited on its own. And finally, a developer must be physically present in the development environment to take advantage of the lava lamps and the information they offer. This makes them not applicable for distributed teams (unless they are installed at each site).

BuildBot's popularity among some developers and observers may be due to the fun and the novelty factor. One developer noted that "*I liked the robot because it typically made a bad situation fun. Breaking a build is never a fun thing, because you know you have to fix it, and it sort of added a playful edge to it.*" Another felt that despite the distraction, BuildBot was fun and "*a positive thing to relieve your stress.*"

Some participants felt that the robot was one of many distractions in the lab, and reacted by listening to music. Others, however, did not like the fact that the robot notifies everyone in the room, whether or not the notification applies to them. As one observer expressed, "*it is distracting. If it's in the far corner, that's like 5 minutes to walk there and 5 minutes to walk back, and you can hear the motors throughout the entire [environment] the whole time…*"

Additionally, some developers expressed concern about the robot's singling out of one single developer. While one observer described this as "*a healthy embarrassment*", we believe this has the potential to cause some developers stress and might negatively impact the team.[1]

There were also concerns about the reliability of the build scripts used. One developer voiced suspicions about trusting notification systems: "*in my experience with Ant scripts, they're so flaky that I have a hard time trusting that everything went to fruition 100% of the time. Just because the robot's not moving, doesn't mean the build status is 100% correct.*" The developer also added that "*I don't trust Ant scripts. A lot of the time they can die silently.*"

5 Conclusion

The results of this evaluation show that the social nature of the group must be considered, as well as the preferences of each individual developer, when introducing any kind of notification device into a development environment.

The Java Lava Lamps used in this study were well-received in that they were not distracting and a fun addition to the development environment, but we believe this device would be better suited to a more open environment without cubicle walls.

Any team working together creates a kind of bond, and introducing something as potentially disruptive as BuildBot can break that bond and cause friction within the group. While some developers welcomed a fun distraction, others preferred to work uninterrupted.

[1] While we haven't yet compared BuildBot with a manager approaching the developer after a breakage, we believe that the latter might cause developers more stress than the robot.

The lava lamps in this study were obscured by the opaque cubicle walls, but the developers found the information very accessible when the lamps were visible to them (for example, when they were standing up). Since email was the most popular and the lava lamps were not distracting to the developers, we conclude that the most effective for an agile development group would be a combination of an openly visible but unobtrusive ambient device and a virtual device such as email.

6 Future Work

The results presented here are those of a small-scale, short-term study. A longer-term evaluation (months or years) is needed involving many more developers, preferably in an industrial setting.

There are also many kinds of alert mechanisms that have yet to be evaluated. Ceiling-mounted coloured rope lighting, system tray alerts, a visit from a project manager, or Ambient Orbs [11] can also be used to notify developers to a build failure. The modes used in this study could also be modified; suggestions for future studies include sending emails to all team members, or of a virtual lava lamp that is always visible on the developer workstations when the build is broken.

The work presented here could be further expanded into other areas, where status alerts of different types (sound, visual) or attention levels (ambient, alerts) would be used to notify members of different types of teams.

References

1. *Manifesto for Agile Software Development*, 2005. Accessed 5 June 2007. http://agilemanifesto.org
2. *Anthill Pro* build management system. http://www.anthillpro.com/html/products/anthillos
3. CruiseControl. http://www.cruisecontrol.sourceforge.net (Accessed February 2007)
4. Saff, D., and Ernst, M.D.: An Experimental Evaluation of Continuous Testing During Development. In *International Symposium on Software Testing and Analysis* (*ISSTA '04*), pp76-85, Boston, USA, July 11-14, 2004.
5. Cadiz, J.J., Venolia, G.D., Jancke, G., Gupta, A.: Sideshow: Providing peripheral awareness of important information. *Microsoft Research Technical Report* MSR-TR-200181, 2001.
6. McFarlane, D.: Coordinating the Interruption of People in Human-Computer Interaction. *Proc of the IFIP TC.13 Conference on Human-Computer Interaction* (Interact 2001).
7. Cutrell, E., Czerwinski, M., and Horvitz, E.: Notification, Disruption, and Memory: Effects of Messaging Interruptions on Memory and Performance. *Proc. of the IFIP TC.13 Conference on Human Computer Interaction* (Interact 2001).
8. Savoia, A.: On Java Lava Lamps and other eXtreme Feedback Devices. 26 August, 2004. http://www.artima.com/weblogs/viewpost.jsp?thread=67492
9. Mike Clark's Build Automation Blog. http://www.pragmaticautomation.com/cgi-bin/pragauto.cgi
10. Ablett, R., Sharlin, E., Maurer, F., Denzinger, J., and Schock, C.: "BuildBot: A Robotic Self-Supervision Mechanism for Agile Software Engineering Teams", *Proc. of* IEEE RO-MAN 2007.
11. Ambient Devices' Ambient Orb, http://www.ambientdevices.com/cat/orb/orborder.htm