A Robotic Colleague for Facilitating Collaborative Software Development

Ruth Ablett Shelly Park Ehud Sharlin Jörg Denzinger Frank Maurer
Department of Computer Science
University of Calgary
2500 University Drive NW
Calgary, AB, T2N 1N4, Canada
{ablettr, parksh, ehud, denzinge, maurer}@cpsc.ucalgary.ca

ABSTRACT

Robots exist in both the virtual domain of computers and the physical realm with humans, and therefore offer an effective interface between the two. A robot as an autonomous mobile agent can offer visual, audio and tactile interaction for a team of humans to support computer-mediated communications. In this paper, a robot is used to mediate communication between humans for Agile software engineering teams and also delivers system critical information to the developers by providing ambient information about the software build. We believe that agile software engineering, with its human-centric practices, can benefit from the use of a robot to facilitate collaborative software development, and enhance communication between developers.

Keywords

Collaborative Software Development, Robots, Human-Robot Interaction, Agile Software Engineering, Natural Language Understanding

1. INTRODUCTION

Agile methods refer to human-centric software engineering methodologies that advocate developing high-quality software in short iterations. Agile approaches emphasize interactions and collaborations between people [1] rather than large documentations and rely heavily on automated regression testing to ensure internal software quality. Because the methods emphasize face-to-face interaction and producing working software in short iterations, the communication between the team of developers can be intensive and constantly requires context-sensitive information about the state of the development progress.

A robot has the potential to be an effective assistant to an agile team, especially in supporting face-to-face team communications about the development progress and in providing ambient display about the software build to quickly assess the state of the project, thus providing encouragement or an incentive to improve. The robot is unique in that it possesses the ability to physically respond to virtual stimuli, bringing awareness information from the digital realm into the physical and vice-versa. In this paper, we present two robotic support functionalities for agile teams: BuildBot works in cooperation with humans to help achieve continuous integration and ScrumBot supports project progress meetings (so called daily scrums).

CSCW'06, Nov 4–8, 2006, Banff, AB, Canada.

2. BACKGROUND INFORMATION

In *continuous integration*, every time new code is checked into the shared source code repository, the entire software is re-built, deployed and tested against a suite of automated regression tests. *Continuous integration* and frequent check-ins of tiny increments of code ensures that existing functionality is not broken by the new code. A simple bug which may take only a few minutes to repair in the early stages may end up costing huge numbers of person-hours if not detected early. Continuous integration facilitates the early detection of bugs.

Savoia [3] has created an ambient feedback device, *Java Lava Lamps*, that helps the team keep track of the build status. The continuous integration server is connected to two lava lamps, one green (indicating a stable build) and one red (indicating a broken build). Only one has power at any given time. Because lava lamps take a few minutes to heat up, it was possible to tell how long the build had been broken, judging by the bubbles of lava on both lamps. Further, the developers were trying to fix the problem before the lamp heated up – this voluntary, playful behavior created a self-supervision of developers instead of having to involve a manager. While this approach is simple, it is also limited by its ambient, visual-only nature. The developers must look at the lamps to get an idea of the build status.

In Agile development, the software requirements are written down in a form of index cards rather than in a long paragraphed document. These index cards are used to document requirements and measure the development progress. The purpose of a *daily Scrum meeting* is to briefly communicate the developers' progress, report problems they encountered and discuss the plans for the next iteration.

3. DESIGN AND IMPLEMENTATION

With **BuildBot** (based on a Sony AIBO [4]), we are trying to further the human-computer collaboration by using a robot as a collaborating tool to *actively* deliver ambient information. We believe robotic embodiment of the state of the software build can help an agile team collaborate more effectively, especially if the robot can physically interact with the team members. The robot would act as a *dynamic information radiator* that delivers the information physically rather than a static one that tends to get ignored [2]. When the ambient data is applied to *continuous*

integration, it provides important information pertaining to the current build status and alerts the team when the different parts of the software don't integrate properly. It also gives the team a sense of accomplishment that the project is being tested and introduces individual accountability in a playful way.

The main design goal behind BuildBot was to use the idea of the collective awareness of agile teams to create an engaging and fun tool that will help the team to fix a broken build as quickly as possible. Whenever a change is uploaded to the shared code repository by a team member, the *continuous integration* component runs a script that integrates the entire team's code together. If the new code integration and testing was successful, BuildBot provides positive feedback to the entire team by happily barking from its home base and showing green LED lights.

If the tests fail, the build is broken. Then BuildBot would deliberately walk slowly to the individual who had uploaded the new, broken code and display to the team that it is unhappy with that person. BuildBot will deliberately walk slowly and dramatically to alert the team of the broken build via sound and visual cues, and to the responsible individual through an e-mail, alerting and giving them the time to fix the problem. It also creates a kind of playful tension as the other team members wonder where the robot will be going. Giving the responsible individual a lighthearted and friendly 'punishment' introduces more targeted accountability.

In order to allow the robot to walk to the team member's desk, we designed a vision algorithm analyzing the streaming video from the robot's camera. For simplicity, white tape was used for the lines on the floor leading to each team member's desk. These lines are a navigation guide, linking BuildBot's base station to the network of lines and allowing it to walk to a developer's desk via the simplest route. The lines on the floor have junctions which branch off at 90 degrees.

When walking, the robot keeps track of these junctions and consults an internal map which gives directions on how to get to each workstation based on the junctions it encounters. Once BuildBot reaches the end of a line, it looks up and gently 'punishes' the team member by barking and growling. This robotic reprimand will cease when the build is fixed, or when the robot senses a touch on its head sensor. We currently have a working prototype of BuildBot which was evaluated only in an informal and limited user study.

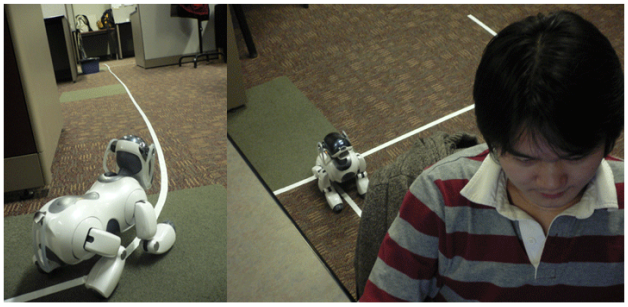

Figure 1: BuildBot delivering the message of broken build

ScrumBot participates in *daily Scrum meetings*. The goal of this project is to further computer-mediated collaboration based on human speech. ScrumBot is collaborating with the group by gathering information and distributing it to the team. Currently the system has two separate parts: the summarizer engine and the robot's emotional engine. The summarizer is made up of a list of basic phrase lists and a list of important key words. The speech recognition engine [5] is fed with the basic set of phrases compiled from previous meetings and phrases in index cards. Research has shown that instead of solely relying on the speech recognition, which generally produces many incorrectly recognized phrases, confining the system to listen for some pre-chosen phrases can improve the recognition rate [6]. When pre-chosen phrases are heard, they and their context are compared with the list of important keywords. The summarizer extracts only the phrases that contain the important keywords and publishes a meeting summary. In the current state, the meeting is facilitated by the system as it currently cannot deal with ad-hoc meetings and cannot gracefully recover from unpredictable scenarios of human conversations.

Currently, the robot can express two states of emotions and it can interact with humans tactical senses on the table. If the robot is happy it expresses this emotion to the person that made it happy (by contributing what was expected from this person) by looking at the person and displaying nice looking colors. It can also express confused state by walking away from the "offending" person. The already mentioned vision algorithm makes sure that the robot will not walk outside the table boundaries.

4. FUTURE WORK

Our major future effort for this project is to perform a formal evaluation of our system with a group of developers and to determine if BuildBot and ScrumBot are actually helping the team. BuildBot in its current implementation is not able to recharge its own batteries. BuildBot should be able to find its own power station using its camera. We are also need to address the issue of BuildBot's inability to recognize if a developer is actually sitting at her workstation. For ScrumBot, we are planning to integrate the summarizer engine with the physical robot interface in the near future. The robotic interface emotional engine has to be tightly integrated with the confidence level of the speech recognition. The next major step is to let humans facilitate the meeting when the robot behaves only as an assistant that summarizes the meeting. Fulfilling this still requires extensive system training and user evaluation to improve both the speech recognition and user interaction.

5. REFERENCES

[1] Agile Manifesto. http://agilemanifesto.org

[2] Cockburn, A. *Agile Software Development: The Cooperative Game*, Agile Software Development Series, Addison-Wesley, NJ, 2001, pp 70-80

[3] Savoia, A. "eXtreme Feedback for Software Development", 2004 http://www.developertesting.com/archives/month200404/20040401-eXtremeFeedbackForSoftwareDevelopment.html

[4] Sony Aibo Home page. http://www.sony.net/Products/aibo

[5] Microsoft SAPI http://www.microsoft.com/speech/download/sdk51/

[6] Park, S., Denzinger, J., Maurer, F., Sharlin, E. "An Interactive Speech Interface for Summarizing Agile Project Planning Meetings", CHI 2006 WIP, ACM Press, 2006