Presence Awareness in Global Software Engineering

Anders Bech Mellson
anbh@itu.dk

Kristian S. M. Andersen
ksma@itu.dk

Mads D. Christensen
mdch@itu.dk

Author Keywords
Activity Monitoring

General Terms
Project proposal, Documentation, Theory

BACKGROUND

Nowadays, globalisation is an economical and societal trend that has pushed industries to move from local to global markets requiring practitioners to work more and more in distributed arrangements. However, this rapid shift of practice has not been followed timely by organisational changes. In fact, a fundamental difference between a co-located arrangement and a distributed one is represented by the lack of physical presence. Often underestimated, this intrinsic characteristic has severe repercussions, which need to be properly addressed for the distributed cooperation to succeed. Paraphrasing Herbsleb [1], many of the mechanisms that function in a co-located setting are absent or disrupted in a distributed project. Different approaches have been investigated to improve the awareness of the working context that a member of a virtual team has; nonetheless, information like the presence of virtual team members, trivial in a co-located setting, represent an interesting open area of investigation.

IDEA

We wish to create a presence information infrastructure for global software engineers. The central idea behind the project is focused on proposing a solution to the context awareness problem described in the background. Our initial conceptualisation of a possible implementation of such solution considers a modular system able to gather presence information from inputs sources (sensors), which would them be exposed via a REST API. Since there could be many workers using this system, each having many sensors and consumers (demonstrators), the system needs to be highly scalable. For this reason we will create a system based on the Actor model by Carl Hewitt [2]. The infrastructure holds the current state of each worker's presence, obtained through the sensors. We will use this information to create a number of demonstrators, showing possible applications translating the presence information into awareness and availability.

[1]Herbsleb, James D. (2007). Global Software Engineering: The Future of Socio-technical Coordination

[2]Carl Hewitt; Peter Bishop; Richard Steiger (1973). A Universal Modular Actor Formalism for Artificial Intelligence

SCENARIO

In our scenario we look at a small danish Software Development company called Robocat located in central Copenhagen. The company has 11 employees where 7 employees work in the Copenhagen Office, 2 in Aarhus and 2 in San Francisco.

Due to the distributed nature of the company it is very hard to gain an overview of the people present at work and whether their presence is in the office or somewhere else. Workers are teamed up on projects that span multiple locations and often need to coordinate ad hoc meetings.

In this scenario the collective knowledge about coworkers presence could alleviate some of the issues that occur from the global distribution of teams. The teams could use our infrastructure to collect presence about all workers and feed it to demonstrators that would help the team to gain better insights into the collective status of the team. The sensors that detect their presence is feeding the infrastructure with information through the REST API, describing their individual situation. These sensors could be:

- The proximity of a team members phone to their work computer.
- An application installed on the team members computer that detects work-related activities.
- A camera monitoring movement patterns in the room where the work is executed.
- A heat sensor/thermal camera detecting movement in the work area.
- A pressure plate on the team members chair.
- A RFID chip that the team member must log in with.
- A bot logged in to the teams work chat room watching team members online status

Each of the team members could have an application on their workstation (one of the demonstrators), exposing availability information of the other team members. The program calls the service through the REST API and retrieves information about the other team members. Based on this information, it determines whether each team member is present and available. Examples of demonstrators in this scenario could be:

- An application installed on the team members mobile device that displays the status of all team members.

- USB toy figures conveying presence information of team members.
- A service that detects when the team member tries to contact another team member. The service will then inform the interrupter about the status of the interruptee.
- An application that can visualize presence information over time and use it for statistical purposes.
- A large wall mounted display that can convey information about many team members.

All of this information can be accessed by anyone within the company, and can be used to get a quick overview of how the companys resources are being used. It can also be used by a developer inside the company to see where another employee in charge of a given area is currently located and if that person is busy.

REQUIREMENTS

Server for the infrastructure. We will use Azure, Digital Ocean or similar. Sensors for the presence information gathering. We will consider using some of the following; smartphones, heat sensors, cameras, microphones, RFID sensors, pressure sensors, software sensors. The infrastructure should be able to handle data coming from multiple sources even when carrying conflicting information. Demonstrators showcasing the infrastructure. Such as mobile devices, screens, holographic displays, speakers, indicator lights, wall-mounted displays.

SUPERVISORS

Paolo Tell and Jakob Bardram

FORESEEN OUTCOME

The outcome of this project will end up with the following products.

1. (prototype) infrastructure. The server software system supporting the entire project
2. (prototype) REST APIs. The software exposing the functionalities provided by 1.
3. (prototype) demonstrators e.g. actuators that uses the information from the infrastructure.
4. (report) infrastructure design. Internal document detailing all design decisions, diagrams, etc related to 1. This document will be used during the execution of 1 to collect all the requirements and create the design of the infrastructure
5. (report) APIs documentation. Formal documentation of the functionalities of 1. exposed
6. (report) input sources. Internal document capturing all the discussions including rationale related to the HW/SW considered for sensing the presence. The document will contain also a summarizing matrix.
7. (handout) final report. Formal document for the course in the format of a Ubicomp paper.

RISK MANAGEMENT

If we run into an unexpected complication we have assessed that the minimum work we willhandin is the infrastructure prototype and the Ubicomp report.

LOGISTIC

- We have a weekly meeting on every tuesday. During this meeting:
  - Every team member is available on Slack and Skype.
  - We discuss the work progress during the last week.
  - Assess when we have the need for an alignment meeting with our supervisor.
  - Follow up on the schedule.
- We have a shared folder on Google Drive which is accessible by team members and the supervisor.
- We use GitHub to version and ensure our prototypes and report.

PLAN WITH WORK PACKAGES

- Overall Information
  - start week 37 - 09.09.14
  - duration 14 weeks - 1 week vacation
    - Infrastructure 5 weeks
    - Demonstrators 4 weeks
    - Report 4 weeks
- Infrastructure prototype
  - This package comprises the design and implementation of the infrastructure. Following the tasks and deliverables.
  - start week 37 - 09.09.14
  - duration 4 weeks
  - Tasks
    - Requirement identification
    - High level design
    - Technical description
    - Implementation
    - Test
  - Deliverables
    - Initial design
    - Technical description
    - Final design
- Demonstrator prototypes
  - Design and implementation of a few demonstrators that will use the infrastructure to create awareness and availability information.
  - start week 43
  - duration weeks
  - Tasks

    - Brainstorm potential demonstrators
    - Select and decide which demonstrators to create
    - Technical description
    - Implementation
    - Test
  - Deliverables
    - Initial selection of demonstrators with descriptions
    - Design of each demonstrator
    - Technical description of each demonstrator
    - How-to guide for each implemented demonstrator

- Report
  - Report written using the Ubicomp template describing the infrastructure, looking at similar work, evaluation of the infrastructure and showcasing a few demonstrators.
  - start week 48
  - duration weeks
  - Tasks
    - Write outline
    - Find related work
    - Create figures and technical illustrations
  - Deliverables
    - Initial report outline
    - Figures and technical illustrations
    - Final report outline
    - Report drafts
    - Final report

RELATED WORK

J. Herbsleb. Global software engineering: The future of socio- technical coordination. 2007 Future of Software Engineering, 2007.

J. D. Hincapi-Ramos, S. Voida, and G. Mark, "A Design Space Analysis of Availability-sharing Systems," presented at the Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, 2011.

I. Steinmacher, A. Chaves, and M. Gerosa, Awareness Support in Distributed Software Development: A Systematic Review and Mapping of the Literature, Computer Supported Cooperative Work (CSCW), vol. 22, no. 2, pp. 113-158, 2013.

S. Houben, S. Nielsen, M. Esbensen, and J. E. Bardram, NooSphere: An Activity-centric Infrastructure for Distributed Interaction, presented at the Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia, New York, NY, USA, 2013, pp. 13:1-13:10.

Y. Dittrich, D. Randall, and J. Singer, Software Engineering as Cooperative Work, Computer Supported Cooperative Work (CSCW), vol. 18, no. 5, pp. 393-399, 2009.

K. Schmidt, Cooperative work and its articulation: requirements for computer support, Le Travail Humain, pp. 345-366, 1994.

Jacob B. Cholewa and Mathias K. Pedersen, ocon "a context aware framework, 2014