Introducing Tempest, a Modular Platform for In Situ Data Collection

Nikolaos Batalas
Eindhoven University of Technology
Den Dolech 2, 5600MB
Eindhoven, The Netherlands
n.batalas@tue.nl

Panos Markopoulos
Eindhoven University of Technology
Den Dolech 2, 5600MB
Eindhoven, The Netherlands
p.markopoulos@tue.nl

ABSTRACT
In this paper we present Tempest, a tool for conducting studies that rely on in situ data collection, such as the Experience Sampling and Diary methods. It implements a modular architecture and relies on interchangeable parts to allow for increased suitability for multiple purposes and long-term reliability. Tempest focuses on ease of use and deployment, and on making use of participants’ own devices.

Author Keywords
In situ data collection, Mobile Sensing

ACM Classification Keywords
H.4.m Information Systems Applications: Miscellaneous; J.4 Computer Applications: Social and Behavioural Sciences

INTRODUCTION
The class of studies that aim to collect data from participants in real situations, as their lives unfold, (e.g. the Experience Sampling Method, or Diary Studies), offer various desirable advantages for researchers, when compared to methods that lack their temporal and spatial immediacy. They bring increased ecological validity to a study, as well as mitigate memory biases in self reports.

Consumer-level mobile devices capable of general purpose computing have become powerful, affordable, and ubiquitous enough, to attract the interest of researchers as suitable platforms to conduct such studies with, moving away from the more cumbersome paper-based solutions of the past. The development of software for the purposes of these research methods has repeatedly attracted academic interest, and a consensus exists as to what the functional requirements of these applications should be[5]. In general terms their aims are to facilitate effective information management for both researchers and participants, simplify the deployment and monitoring of the study, minimize monetary expenses for the researchers.

BACKGROUND
Successful tools for implementing experience sampling or similar methods have been developed in the past and still remain in use. However, the fast evolving nature of mobile platforms has rendered many software development efforts obsolete, in terms of their relevance to the current technological landscape. Notably, ESP[2], one of the most widely used tools for experience sampling, still in use today, runs on Palm PDAs of the early 2000s. Another tool which has known extensive adoption, MyExperience[4], has been built for the Windows Mobile operating system, now officially discontinued. This clashes with stated goals for such platforms, such as making use of participants’ own devices, or collecting rich data through sensors available in modern hardware. More recent tools, even if they can be readily adopted today, are also susceptible to obsolescence.

Additionally, data collection tools only manage to cater to specific fragments of particular needs, depending on the relevance of their hardware platforms and the willingness of their developers to sustain them. The situation is further aggravated by the fact that each tool stands insulated, developed in isolation from similar efforts. This has led to a very fragmented landscape of tools, and recurring incarnations of platforms, where only incremental or partial implementation would be needed.

In response to this issue, we have proposed, in past work, considerations to be taken into account during the design and development of such tools. which we consider collectively under the umbrella of tools for in situ data collection[3]. In this paper, we outline the choices that informed our design and present an implementation that can be put to use for a variety of purposes with regard to studies in the wild.

ARCHITECTURE
The system, illustrated in Figure 1 consists of a modular stack that aims to keep each component insulated from others, for maximum interchangeability. Central to this concept is the browser component as a dynamic execution environment, which separates client-specific low level functions from higher level logic and user interface components, which are executed in the browser’s environment. In this way, a separation of concerns is enforced, and only specific modules need to be catered to for extensibility and interoperability of the system’s components.

Moreover, the proposed modularization allows the use of established codebases of libraries that are used widely and are
FEATURES
The showcased system allows the management and configuration of screens to be delivered a participant’s device. Screens can be any html5 compliant scripts and javascript code. The system comes preconfigured with screen objects of question-answer widgets that pertain to typical experience sampling and diary configurations. A generic configuration GUI is also available that can help a researcher set the properties of screen objects, if those properties have been declared in a specific way by the object’s programmer. Additionally, spatial and temporal triggers can be set that will instruct clients to present the screens to the participants when their respective conditions are satisfied. The researcher is able to view responses by the participants and modify their screens in near realtime.

Configurations are stored in a server, and the resulting screens can be accessed by participant clients. In the simplest scenario, these can be regular browser applications, in which case the participant simply interacts with a website. In more complex scenarios, a dedicated client can access the stored configurations and take appropriate action, e.g. instantiate screens based on a clock trigger at an appropriate time and signal the participant for interaction, or in the absence of an internet connection, store the participant’s answer locally, until an internet connection becomes available. So far the system features an android and a desktop client, which runs on linux, macosx and windows.

The system makes use of established codebases with extensive community support, and allows developers to make use of libraries of their own preference. They can either make use of facilities it already offers or completely ignore them, to whatever extent they find desirable.

CONCLUSION
Our implementation allows different in situ data collection protocols, traditionally considered separately, to be treated collectively under the programmed behaviour of an application stack. Not only ESM and diary methods can be implemented, but also in situ prototyping and user tests, as long as the corresponding screen objects are put into place.

The separation of concerns that is enforced, allows for interchangeable components to be catered to by their respective problem-owners. A social scientist need only bother with a configuration GUI, while a systems developer can implement a new client as long as they can feature a web-view facility[1], without having to reimplement interface elements.

It is our belief that this sort of modularity makes a contribution to a class of in situ data collection platforms that feature interchangeable and interoperable components, thus easing the onerous burden of keeping up with the latest features and requirements for the development of such systems on both the rapidly evolving mobile devices, as well as desktops.

REFERENCES