A Graphical DSL to Design Evolvable Citizen Observatories*

Kennedy Kambona, Jesse Zaman, Wolfgang De Meuter
Software Languages Lab
Vrije Universiteit Brussel
Brussels, Belgium
{kkambona,jezaman,wdmeuter}@vub.ac.be

***Abstract*—Citizen observatories are community-based data-gathering systems that utilize various technologies to process data from citizen science campaigns. Building citizen science applications to support these observatories is a complex endeavour as it requires knowledge of several software engineering techniques: distribution for modelling client/server-side interaction, programming paradigms to implement functionality in different programming languages and to enforce reactive processing for real-time monitoring and feedback, database technology for storage and retrieval of information, etc. Many such systems are touted as simple and lightweight, and are often intended for single use but require a team of developers to construct – and therein lies their contradiction. In this paper we present DISCOPAR, a visual reactive programming language that allows specialists and end-users alike to build evolvable and reusable citizen observatories using a component-based approach. A DISCOPAR program is conceived as a directed-acyclic graph that models the flow of information from client mobile apps to the server-side in a reactive manner. We motivate the evolvability features of DISCOPAR by building five different citizen science applications in conjunction with local citizen science groups in Flanders.**

***Index Terms*—visual programming, DSL, citizen science**

I. Introduction

Citizen observatories are distributed, community-based data-gathering systems that employ various tools and technologies to collect, process and distribute collected data. The reference architecture for these systems typically consists of three main elements. The first element is a suite of client-side applications that allow the upload of some form of data, e.g., images, GPS coordinates etc., from a data source to the server infrastructure, which forms the second element. The server-side logic contains processing elements that perform data extraction, preprocessing and transformations to extract value from the data. The server also stores the data for subsequent processing or analyses. Lastly, a third element s consists of a "dashboard" where key information is presented to the user with control functions for visualising and analysing processed data in various ways. For example, an app for multiple users to count birds in a city will require uploads of individual counts, locations and images, processing for aggregating the counts and a visualisation for showing different counts in different neighbourhoods.

This work is funded by the FLAMENCO project of the Flemish Institute for Innovation by Science and Technology.

Building these types of systems is a complex endeavour, as it requires knowledge ranging from distributed techniques for programming client and server-side interaction, using multiple programming languages, as well as knowledge of database technologies. Furthermore, modern applications supported by such systems have evolved from their delayed, static processing demands to having more dynamic, reactive processing for immediate feedback. For example, organisers can monitor participation by users counting birds, in real time. This means that the server-side logic should support some form of reactive semantics in order to process received data instantly e.g., for delivering instantaneous feedback. Programming these semantics further contributes to the complexity of developing such applications.

Many such systems are touted as "low budget systems," and therein lies their contradiction. They are constructed for a customer such as an institution (a company, municipality) or a societal group (a local biking club, grassroots organisations) with the intention of gathering data from users or citizens in order to draw conclusions about a particular concern. They often require a "small, lightweight application" in order to accumulate and visualise some data for a particular situation. This is an example of the "app paradox", where these are often single-use systems for which there is little budget but require a team of developers to construct.

In this paper, we present DISCOPAR, a visual reactive programming language that allows specialists and end-users (i.e., non-ICT experts) alike to build reusable and evolvable citizen observatories. A DISCOPAR program is conceived as a directed-acyclic graph (DAG) that models the flow of information from client mobile apps to the server-side, and relays results to the mobile apps or the dashboard, in a reactive manner. Evolution is at the core of the construction process of DISCOPAR programs, because the concept of developing them was conceived as a continuous co-creation process between the users and the developers. We present the DISCOPAR language and motivate its evolvability features by building five different citizen science applications and enacting their campaigns, in conjunction with local citizen science groups and institutions in Flanders.