SCIENCES & BIOENGINEERING SCIENCES

The Research Group

Software Languages Lab

has the honor to invite you to the public defense of the PhD thesis of

Sam Van den Vonder

to obtain the degree of Doctor of Sciences

Title of the PhD thesis:

On the Coexistence of Reactive Code and Imperative Code in Distributed Applications: A Language Design Approach

Promotors:
Prof. dr. Wolfgang De Meuter (VUB)

The defense will take place on
Tuesday, June 21, 2022 at 17h in auditorium I.2.03

The defence can be followed via a livestream. Please send an email to Sam.Van.den.Vonder@vub.be for more information.

Members of the jury

Prof. dr. Viviane Jonckers (VUB, chair)
Dr. ir. Roxana Rădulescu (VUB, secretary)
Prof. dr. ir. Kris Steenhaut (VUB)
Prof. dr. Coen De Roover (VUB)
Prof. dr.-ing. Mira Mezini
(Technische Universiteit Darmstadt)
Prof. dr. Tijs van der Storm
(Rijksuniversiteit Groningen)

Curriculum vitae

Sam Van den Vonder obtained his Bachelor degree in Computer Science in 2014 and his Master degree in Computer Science in 2016 at the Vrije Universiteit Brussel, magna cum laude. Afterwards he started his PhD at the Software Languages Lab (SOFT) under the supervision of Prof. Dr. Wolfgang De Meuter. He contributed to the interdisciplinary "Flamenco" project by Flanders Innovation & Entrepreneurship (VLAIO), and he obtained a PhD fellowship "strategic basic research" from the Research Foundation - Flanders (FWO). He coauthored 6 international workshop publications (2 as first author), 1 international conference publication (1st author) and 3 international journal publications (1st author). His work was presented at international conferences and workshops to a scientific audience. He guided 4 bachelor thesis students.

Abstract of the PhD research

The continued increase in computing power and network resources has changed the landscape of software development. Nowadays, software is not confined to a single computer, and instead runs on many types of devices that interact on an *open network*, which is a network that various types of devices such as smartphones, smartwatches, and Internet of Things devices can spontaneously join and leave. Moreover, their software is continuously interacting with each other, e.g., a smartphone smart home application is updated in real-time based on the connected smart home sensors. In other words, the software that powers those devices has become *reactive*.

Existing reactive programming techniques exhibit 3 issues that make them difficult to apply to software for open networks. The first issue is related to *responsiveness*, where data received via a network can (accidentally) make the program *no longer reactive*. Second, we describe a *paradigmatic mismatch* between reactive code and imperative code, both of which are present in every reactive application. The final issue relates to the *open network*, where existing reactive programming techniques have little or no support to *discover devices* on the open network, and to manage their presence in the reactive program *correctly and efficiently*.

We propose a novel computational model, called the "Actor-Reactor Model", that avoids the issues by using so-called "actors" and "reactors" to build distributed applications. First, reactive code is encapsulated by reactors which guarantee responsiveness. Second, imperative code and reactive code is strictly separated in actors and reactors respectively, thereby avoiding a paradigmatic mismatch. Finally, we design a mechanism to discover actors and reactors on an open network, and to maintain their presence correctly and efficiently throughout a distributed reactive program.

The Actor-Reactor Model was implemented in a new programming language called Stella, which serves as a linguistic vehicle to demonstrate the ideas of our approach. Compared to other reactive programming languages and frameworks, we found that actors and reactors are highly suitable to develop distributed applications. Furthermore, they demonstrate a correct and efficient approach to define distributed computations that involve multiple discovered devices (e.g., sensors). All Stella code presented in the dissertation is executable.

The Actor-Reactor Model leads to a better understanding of the coexistence of reactive code and imperative code in Stella and existing reactive programming languages and frameworks.