



Simplifying software complexity

Professor Serge Demeyer is the spokesperson for a research group that provides solutions associated with the ever-increasing complexity of software intensive systems. Below, he discusses his research interests and elaborates on what he considers the impact of the group's work will be in the future

Could you begin by telling us about your core interests in software engineering?

I am a professor in Software Engineering at the University of Antwerp. Over the past 20 years, my research interests have been extremely diverse, from software evolution and software quality, through to hypermedia. However, the universal theme has always been to help people 'out there' and, since software has such an impact on society, I want to leverage it to improve the world we live in. That's why, ultimately, I found my calling in software engineering – the research field that investigates ways to improve how we build software systems.

Traditionally, software engineering was mainly concerned with reducing the number of 'bugs' inside a software system. In recent years, this focus has veered towards 'agility' – the capacity to respond to changing customer requests, legislation and technology. Additionally, all the social facilities are now part of the software engineering tool suite. My core interests can be summed up in three words: quality, change and social.

What are the key objectives of the Antwerp System Modelling (ANSYMO) research group at the University of Antwerp?

We are a research group investigating foundations, techniques, methods and tools for the design, analysis and maintenance of so-called software-intensive systems. Indeed, our world and society are shaped and governed by systems and software; all devices, machines and artefacts surrounding us incorporate software to some extent. It is the primary driver for all of today's technology.

The nature of software-intensive systems, however, has changed considerably in the past few years. First, the availability of more computational resources, including parallel computation and interactive behaviour has enabled one to tackle ever

more complex applications. Second, the need to consider interactions of software with physical components has led to the study of hybrid systems, adding even more complexity. Third, the view that a software system is a static entity has given way to the view that software needs to evolve – that changes in requirements of platforms can be accommodated easily.

How have you overcome challenges faced in automation in an industrial continuous integration environment?

The main problem is the accidental complexity that comes with deploying a tool in a realistic environment; there is an enormous amount of detail that needs to be taken care of. Each detail looks harmless on the surface, but can be the make-or-break factor for being adopted by practitioners. For example, academic research typically focuses on the precision and recall of the tools. However, when validating in a realistic environment, we noticed that the real problem is actually the amount of data needed to feed into the tool before it can give a good enough recommendation. If a software team works in sprints of six weeks, they will not produce enough defects to train the underlying algorithms.

To deal with the accidental complexity, we rely on an extensible source-code meta-model to represent what's inside a software system that can also be extended to hook in plug-ins when needed. This is not a new idea – the quest for a good source-code meta-model has been going on for over 15 years.

Can you reveal more about this quest for a source-code meta-model?

It all began with the FAMIX meta-model, which originated at the University of Bern in the context of a European project that took place between 1996 and 1999. The project focused on methods and tools to analyse and detect design problems in object-oriented

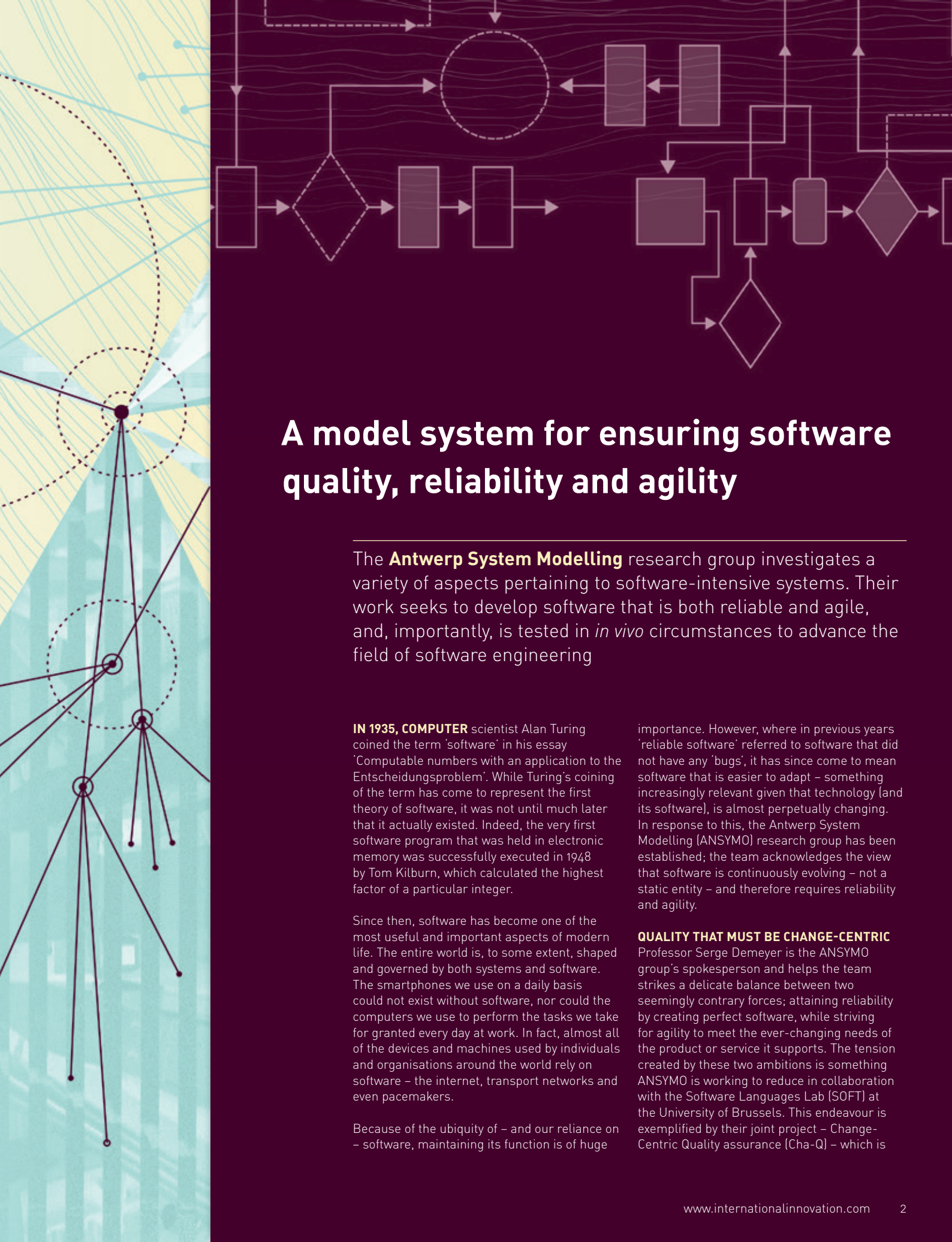
legacy systems. Thus, we needed tools that could manipulate systems in C++, Ada, Smalltalk and Java, where extra analysis tools, such as metrics and refactoring, could be plugged in when needed. In retrospect, this plug-in architecture was a defining feature of the meta-model, and one of the reasons FAMIX was adopted in many other reengineering-related tools. The change-distiller in particular was created at the University of Zurich and, during a six-month sabbatical, I witnessed first-hand how FAMIX influences tool-orientated research in other labs.

How do you envisage the ANSYMO research group will impact software engineering, particularly within industry, in the near future?

The next step for us is to apply our ideas in other areas with stringent quality standards. In particular, in the avionics and automotive industries, where there is a small but thriving ecosystem of SMEs in Belgium. If we could improve the effectiveness of testing in such environments the investment would pay off enormously, because of the risk that defects escape into the field.

We are also seeking ways to apply our techniques in the context of enterprise information systems. Among others, we have contacted the software team responsible for 'tax-on-web' – the Belgian online system where law-abiding citizens submit their tax forms every year. Here, as well, the risk of defects is enormous, so even small improvements can have a big impact.

Finally, we are considering the launch of a spin-off company that offers a combination of tooling (with some form of open-source freemium licensing), with consultancy services.



A model system for ensuring software quality, reliability and agility

The **Antwerp System Modelling** research group investigates a variety of aspects pertaining to software-intensive systems. Their work seeks to develop software that is both reliable and agile, and, importantly, is tested in *in vivo* circumstances to advance the field of software engineering

IN 1935, COMPUTER scientist Alan Turing coined the term 'software' in his essay 'Computable numbers with an application to the Entscheidungsproblem'. While Turing's coining of the term has come to represent the first theory of software, it was not until much later that it actually existed. Indeed, the very first software program that was held in electronic memory was successfully executed in 1948 by Tom Kilburn, which calculated the highest factor of a particular integer.

Since then, software has become one of the most useful and important aspects of modern life. The entire world is, to some extent, shaped and governed by both systems and software. The smartphones we use on a daily basis could not exist without software, nor could the computers we use to perform the tasks we take for granted every day at work. In fact, almost all of the devices and machines used by individuals and organisations around the world rely on software – the internet, transport networks and even pacemakers.

Because of the ubiquity of – and our reliance on – software, maintaining its function is of huge

importance. However, where in previous years 'reliable software' referred to software that did not have any 'bugs', it has since come to mean software that is easier to adapt – something increasingly relevant given that technology (and its software), is almost perpetually changing. In response to this, the Antwerp System Modelling (ANSYMO) research group has been established; the team acknowledges the view that software is continuously evolving – not a static entity – and therefore requires reliability and agility.

QUALITY THAT MUST BE CHANGE-CENTRIC

Professor Serge Demeyer is the ANSYMO group's spokesperson and helps the team strike a delicate balance between two seemingly contrary forces; attaining reliability by creating perfect software, while striving for agility to meet the ever-changing needs of the product or service it supports. The tension created by these two ambitions is something ANSYMO is working to reduce in collaboration with the Software Languages Lab (SOFT) at the University of Brussels. This endeavour is exemplified by their joint project – Change-Centric Quality assurance (Cha-Q) – which is

sponsored by the Flemish research funding institution IWT, and concerned with transferring state of the art into state of the practice.

"In the Cha-Q project, we want to make a leap in current software-engineering tooling by making changes themselves explicit in the backbone of the tools," explains Demeyer. "Indeed, if you look at modern version control systems such as GitHub, software engineers tend to commit much more frequently – we want to exploit this trend." Thus, instead of using a passive storage system for the data, the team wants to develop a means of applying data analytics on the massive amount of data produced within a software team. In doing so, insights into the past, present and future of a particular project are made possible. Ultimately, the team will investigate how to safeguard the quality of a system in an incremental manner, so reliability is maintained as the system continuously improves to meet the evolving needs of what that particular software serves.

A PLETHORA OF PROBLEMS ADDRESSED

In addition to the Cha-Q project, ANSYMO addresses a wide range of other problems. Work is concerned with demonstrating that test processes of software meet quality guidelines, so that, for example, every bug fix is covered by a regression test. Importantly, rather than running all tests for any given release, the team works to identify the tests that are affected by a given change, thereby saving valuable time in determining the precise location of a bug.

Further examples of problems ANSYMO tackles include: ensuring all severe bugs are fixed prior to a release of software, determining who the best person in the team is for handling the issue (as well as reliably estimating the time it will take to solve the problem) and monitoring all changes as they are made, as well as enabling accurate tracking and traceability so that all related code is changed accordingly too. It is not a case of reinventing the wheel each time new software is released or new bugs are found; far more effectively, it provides a means of historical analysis to use what was done before to inform what is done subsequently. This ensures adequate procedure as well as saving countless hours of work – something of obvious benefit to all involved.

LIFE-SAVING WORK PLUS IMPORTANCE OF *IN VIVO*

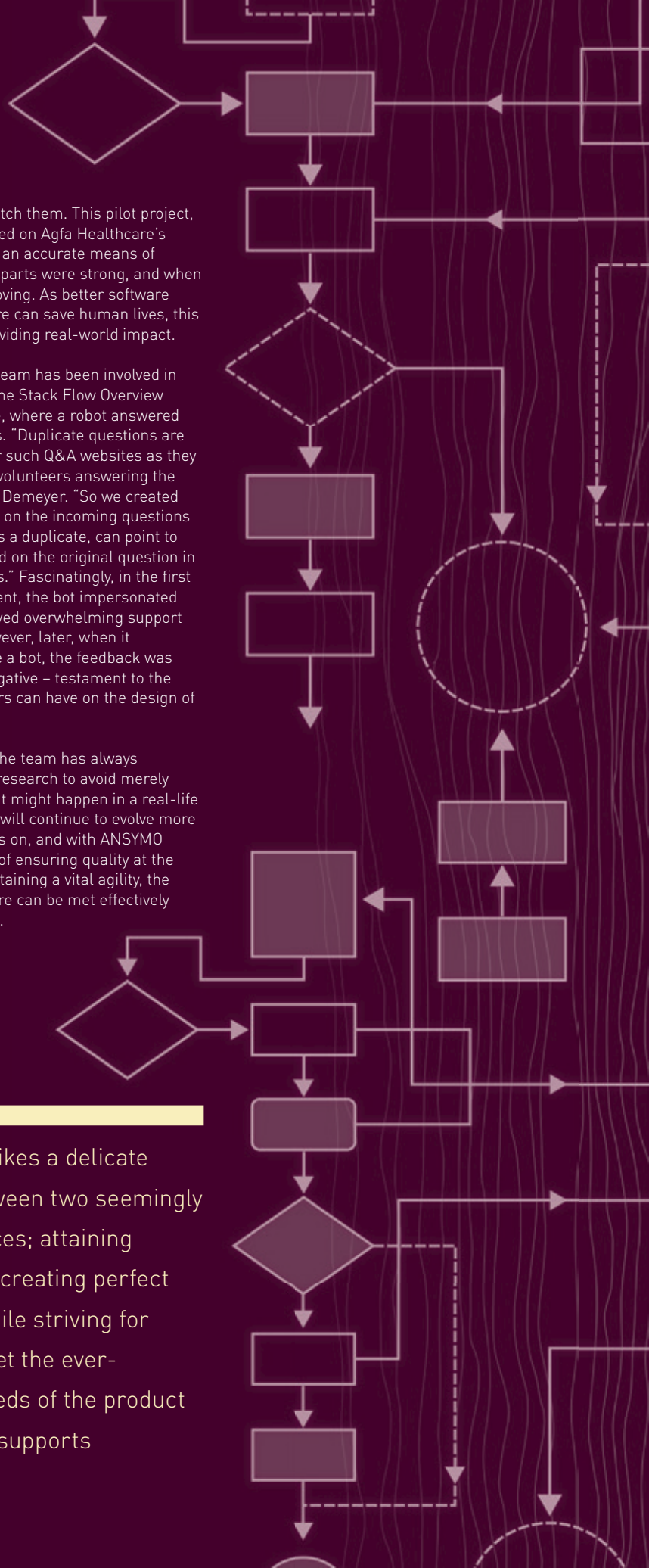
Since its inception in 2010, ANSYMO has made exciting progress in its research on quality assurance. One notable success was its method of injecting artificial defects into 38,000 lines of Java code to verify whether the software

would be able to catch them. This pilot project, which was conducted on Agfa Healthcare's test suite, provided an accurate means of determining which parts were strong, and when parts needed improving. As better software testing in healthcare can save human lives, this methodology is providing real-world impact.

More recently, the team has been involved in an experiment on the Stack Flow Overview community website, where a robot answered duplicate questions. "Duplicate questions are a nasty problem for such Q&A websites as they waste time for the volunteers answering the question," explains Demeyer. "So we created a bot that listens in on the incoming questions and, when it notices a duplicate, can point to the answer provided on the original question in a matter of minutes." Fascinatingly, in the first run of the experiment, the bot impersonated a human and received overwhelming support on the forums; however, later, when it revealed itself to be a bot, the feedback was overwhelmingly negative – testament to the impact social factors can have on the design of software tools.

With that in mind, the team has always conducted '*in vivo*' research to avoid merely speculating on what might happen in a real-life scenario. Software will continue to evolve more rapidly as time goes on, and with ANSYMO providing a means of ensuring quality at the same time as maintaining a vital agility, the demands of software can be met effectively long into the future.

The team strikes a delicate balance between two seemingly contrary forces; attaining reliability by creating perfect software, while striving for agility to meet the ever-changing needs of the product or service it supports



The world is now a numbers game

IN 1985, MANNY Lehman and Les Belady formulated what is known as the 'Laws of Software Evolution'. Two laws in particular are at the forefront of what ANSYMO is trying to achieve through the course of its endeavours.

The Law of Continuing Changes states that a program used in a real-world environment must change, otherwise it will become less useful in that environment.

The Law of Increasing Complexity states that as a program evolves it becomes more complex, thereby necessitating additional resources to preserve and simplify its structure.

Professor Serge Demeyer and his team understand the impact of these two laws, and of how the software community must alter its viewpoints to provide solutions to them. "We are kidding ourselves if we think we can know all the requirements and build the perfect system," explains Demeyer. "The best we can hope for is to build a useful system that will survive long enough for it to be asked to do something new."

In response to this, the software community has embraced the evolutionary view on software so that, rather than seeing it as a problem, they welcome the agility it necessitates. "The agile movement has resulted in a whole new generation of software tools falling under the umbrella of continuous integration or – one step further – continuous delivery," says Demeyer. "Continuous integration allows chrome and Mozilla to release a new version of their browser ever six weeks. Continuous delivery allows Facebook and Google to release several times a day; Amazon even claims to deploy new software into production every 11.6 seconds!"

These remarkable numbers attest to not only how quickly everything is developing, but how quickly companies need to respond and adapt to such developments.

ANTWERP SYSTEM MODELLING RESEARCH GROUP

OBJECTIVES

To investigate foundations, techniques, methods and tools for the design, analysis and maintenance of software-intensive systems.

KEY COLLABORATOR

Professor Coen De Roover, Free University of Brussels, Belgium

FUNDING

Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT)

CONTACT

Professor Serge Demeyer


Spokesperson for the Antwerp System Modelling (ANSYMO) research group

University of Antwerp
Department of Mathematics and Computer Science
Antwerp, 2020
Belgium

T +32 326 539 08

E serge.demeyer@ua.ac.be

www.uantwerpen.be/ansymo

 www.researchgate.net/profile/Serge_Demeyer

 www.be.linkedin.com/in/SergeDemeyer

 www.plus.google.com/+SergeDemeyer



SERGE DEMEYER is a professor in the Department of Mathematics and Computer Science at the University of Antwerp, Belgium, and Spokesperson for the ANSYMO research group. He directs a research lab called Lab On REengineering (LORE), which investigates the theme of software reengineering. He completed his MSc in 1987 and PhD in 1996, both at the Free University of Brussels. His main research interest concerns software reengineering, more specifically the evolution of object-orientated software systems.