



Content archived on 2024-04-23

Feature Stories - Keep it simple: bring software complexity under control

'Fools ignore complexity. Pragmatists suffer it. Some can avoid it. Geniuses remove it.' The words of pioneering computer scientist Alan Perlis, who would probably be shaking his head in dismay at the complexity of modern software systems. Though they may not describe themselves as geniuses, a team of EU-funded researchers are bringing complexity under control, deploying an alternative approach to software engineering to build systems that are safer, cheaper and more robust.



DIGITAL ECONOMY



© Shutterstock

Software has not only become more complex, it has also become more pervasive: it is in your car, in your electricity meter and power station, in planes, trains, banks and hospitals. It runs critical systems where a single error can have disastrous consequences. And the increased risk of failures due to increasing software complexity is not the only problem, complexity also raises costs: testing software for bugs currently accounts for around half of pre-release spending and as much as 70 % of


post-release expenses.

At the heart of the problem is the fact that traditional software engineering processes are ill-designed to handle the enormous complexity and diversity of modern software systems.

'Look at a modern car, for example, there are hundreds of software elements that come together, from controlling sound quality to cruise control. As more and more elements and more functionality are packed into systems, engineers say they are losing control of complexity and worry they won't be able to provide the quality assurances required,' explains Alexander Romanovsky, a professor of computer

science at Newcastle University in the United Kingdom. 'And this complexity is everywhere and in everything: companies that make cars, trains, planes etc. actually dedicate a lot of their time and resources, perhaps as much as half, to developing software: ultimately all industry is producing software.'

The question is, can industry find a better way to make software, particularly for critical systems?

Researchers involved in the 'Industrial deployment of advanced system engineering methods for high productivity and dependability' ([Deploy](#))  project, supported by EUR 12.4 million in funding from the European Commission and coordinated by Prof. Romanovsky, believe there is. Their approach follows formal engineering methods, which, thanks in no small measure to the team's efforts, are now starting to gain acceptance among companies and engineers as a more efficient, practical way to develop complex software systems. Deploy processes have already been used to develop safety critical software for metro lines and airport shuttle trains.

Unlike traditional software engineering approaches, formal methods are grounded in mathematical modelling and analysis, supporting reasoning at multiple levels of abstraction to enable a systematic engineering flow from requirements specification, through architecture modelling and detail design to implementation, testing and deployment. For the project, the team continued the evolution of 'Event-B', a formal method for system-level modelling and analysis first developed in the EU-funded 'Rigorous open development environment for complex systems' (Rodin) project, and supported by 'Rodin tools', an integrated development environment.

Encouraging engineers to think differently

'For engineers, it's quite a fundamental change. Typically, when they embark on a project, they are accustomed to starting to develop the software, adding to it, expanding it, and then doing a lot of the hard work - implementation, verification, testing - at the end. Formal engineering reverses that: the hard work is at the beginning, starting with an abstract of the system requirements and following a rigorous, logical and mathematical development process,' Prof. Romanovsky explains.

The advantage is that errors are caught early in the development and the rigorous, logical approach means that complexity is kept within predefined limits, all of which reduces the need for testing at the final stages. That ultimately leads to more robust and dependable systems, potentially at noticeably lower cost than using traditional engineering methods.

It sounds like a win-win situation, but there is a key challenge: engineers are typically not mathematicians.

'Formal engineering methods require a knowledge of maths which most engineers accustomed to using traditional approaches don't have, and because the process is so different it also requires a change in their mindset. In addition, it's hard to convince a company that's been doing software development in a certain way for 10 to 30 years that they should do it differently,' Prof. Romanovsky points out.

The Deploy project, probably the largest deployment of formal software engineering at the industrial level ever carried out, has gone a long way to overcoming those hurdles.

Working with project partner SAP, for example, the team implemented formal processes transparently so engineers could continue using domain-specific programming languages to develop business critical software, which were then automatically translated into Event-B, backed up by formal engineering specialists.

With Siemens, where engineers already had considerable experience with formal modelling, Event-B has been used to develop software that is currently in use in the Barcelona and Paris metro systems and elsewhere for train control and signalling systems. With Bosch, the approach was used to develop cruise control and start-stop systems for cars. And with Space Systems Finland, another project partner, the process has been employed for components of the European Space Agency's BepiColombo space probe and for altitude and orbit control systems.

French partner Systerel, meanwhile, is using event Event-B for a range of railway and aerospace systems. The Deploy researchers have set up two spin-off companies - Rodin Tools and Formal Mind - to commercialise the tools and extend deployment. They have also embarked on a follow-up project with EU funding called 'Advanced design and verification environment for cyber-physical system engineering' (Advance).

The successful deployment and continued use of formal engineering methods in the companies involved offers a clear example of strategies for other firms to follow as they seek new ways to deal with complexity.

'We're definitely giving industry food for thought about current engineering methods and the advantages in certain circumstances of switching to a formal approach,' Prof. Romanovsky notes.

The tools, extensive Event-B documentation and support offered by the Deploy partners are also likely to help convince engineers to embark on formal engineering projects.

'In the long run, software systems are only going to keep getting more complex -

formal engineering, as we have shown, is one way to address that problem,' Prof. Romanovsky says.

Deploy received research funding under the European Union's Seventh Framework Programme (FP7).

Link to project on CORDIS:

- [FP7 on CORDIS](#) 
- [Deploy project factsheet on CORDIS](#) 

Link to project's website:

- ['Industrial deployment of advanced system engineering methods for high productivity and dependability' website](#) 

Other links:

- [European Commission's Digital Agenda website](#) 

Related projects

	ARCHIVED
	Industrial Deployment of Advanced System Engineering Methods for High Productivity and Dependability
	DEPLOY
	15 July 2019
PROJECT	

This article is featured in...

RESEARCH*EU MAGAZINE



**Women in science — and
research to improve
women's lives**

Last update: 19 December 2012

Permalink: <https://cordis.europa.eu/article/id/90012-feature-stories-keep-it-simple-bring-software-complexity-under-control>

European Union, 2025