CLEANING UP LEGACY SOFTWARE

Arjan Mooij is a senior research fellow in TNO's ESI - Embedded Systems Innovation research group and works on understanding and cleaning up legacy software. Mooij: 'What fascinates me is the reduction of complex code down to its essence. To an elegant piece of software.'

By Reineke Maschhaupt
Image Istockphoto

It was ESI's high-tech partners themselves who brought their legacy problems to Mooij and his colleagues' attention. Their proprietary software systems have been in development for decades; over time, this software has gradually been expanded and extended. What was once effective had to be done faster or slightly differently a few years later. In the process, developers found themselves with less and less time for the old code and preferred to work around it. Today, nobody knows exactly what those millions of lines of code do.

PEELING THE ONION

Mooij's favourite metaphor for explaining the legacy problem is the onion: if you peel back an onion layer by layer, you end up with tears in your eyes. This is the general feeling of Mooij's industrial partners. Mooij: 'The problem is immense. Most companies cannot afford to stand still to clean up their own code, yet software maintenance is desperately needed. Those on the work floor lose an incredible amount of time to understand what the existing software does. As if that's not enough, they are required to innovate as well.'

ESI works with the industry-as-lab approach. Four days a week, researchers work on-site at the partners' companies in parallel to the normal development process. Mooij: 'First of all, we want to give more insight into a company's software because the developers can immediately benefit from being able to work faster.'

BACK TO BASICS

Mooij: 'The second step – making the code simpler and shorter – is extremely challenging and exciting. Rejuvenation is the most thorough approach to this and makes major interventions possible. In this approach, you don't look at how something is done, but rather at what it should do. You then generate new code for that. At Phillips, we used this technique to make a system component almost three times smaller. By peeling back each layer of the onion, we can observe all of the trends in software from the past ten or twenty years.'

Another frequently-used technique is restructuring, says Mooij. 'Thanks to the restructuring technique, you do not need to know what the system does. With this, we can make structured adaptions to millions of lines of code at once. In doing so, we examine the implementation aspect instead of the essential functionality.'

'The biggest challenge is finding an approach that you can easily customise,' he continues. 'There is no single solution that we can roll out everywhere at the simple push of a button. Each situation requires a specific approach. What makes our high-tech partners unique is that they want to learn from each other. If we develop something for one partner, the others know about it.'