Joining scientific forces on big software

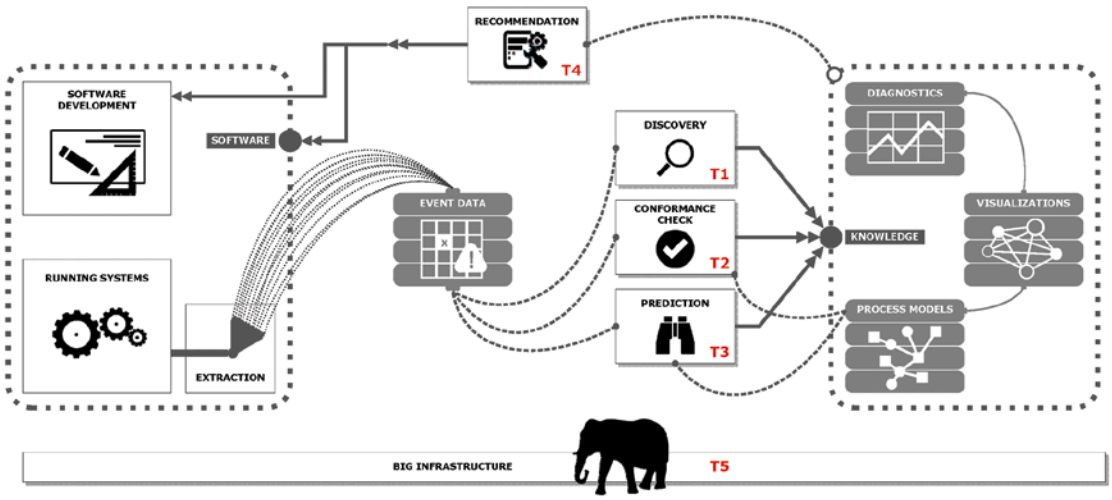
In the 3TU.BSR project, the three technical universities in the Netherlands are researching the run-time analysis of large software systems. The University of Twente's Mariëlle Stoelinga describes the project's approach and goals.

Nour Assay | Vincent Bloemen | Mozhan Soltani | Mariëlle Stoelinga | Gamze Tillem

Software is increasingly like a living organism, evolving constantly and autonomously. Through continuous delivery, continuous deployment and their successor, Devops, software is changing all the time. The software that puts a book in your basket at a web store may have been altered by the time you reach the checkout screen. The key question here is: how do you consistently ensure the quality of such ever-evolving systems?

The project Big Software on the Run (3TU.BSR), a joint endeavour of the Netherlands' three technical universities, focuses on the in vivo analysis of large software systems. Because trends like Devops blur the lines between development and deployment, system validation must shift from testing and verification at design time to monitoring at run time: only at run time do we know the application context and can we carry out the appropriate analyses and – if needed – interventions.

The 3TU.BSR project is all about fully automatic monitoring and diagnosis. By logging all system events and comparing the traces thus obtained with a specification or reference model, we can check system or application correctness: does it behave according to spec? The aim is to develop methods that diagnose system faults automatically,



The 3TU.BSR project follows five tracks, covering the different areas where major breakthroughs are needed: discovery (T1), conformance checking (T2), prediction (T3), recommendation (T4) and infrastructure (T5).

and to synthesize recommended actions: which components need to be debugged, which components need replacing, which components require more testing with what inputs? And when we're better off reconfiguring the system, which components and subsystems should we reconfigure?

Software crashes

The key project ingredient is process mining. Traditionally, there is a gap between model-based process analysis methods such as simulation and other business process management techniques on the one hand and data-centric analysis methods such as machine learning and data mining on the other. Process mining aims to bridge this gap by providing tools and techniques for automated process model discovery, conformance checking, data-driven model repair and extension, bottleneck analysis, and prediction based on event log data.

The group led by project leader Wil van der Aalst at Eindhoven University of Technology is working on new process mining techniques for big software. Researchers are collecting and analyzing data from software executing in its natural habitat and are developing new scalable process discovery methods to infer models that describe the real behaviour of software systems. Through smart visualization techniques, these models can be used to gain insight into what's really going on and where the software did not behave as expected.

As manual model inspection may be costly and error-prone, the group is also developing new conformance checking techniques. These detect deviations by aligning a normative model with actual behaviour (data), enabling us to analyze the software's performance. For instance, we can pinpoint bottlenecks and slow code and the functions or components in which they occur.

Monitoring should not affect program behaviour. This is already a challenge in sequential programming; concurrent software makes it even more of one. Marieke Huisman's research group at the University of Twente will develop a general-purpose approach based on local program annotations and global properties. Runtime monitoring is essential to check concurrent software's conformance during deployment. At the same time, it provides insight into low-level software events, generating a continuous data stream that feeds discovery.

The research group led by Arie van Deursen at Delft University of Technology is using software execution data to automatically reproduce reported software crashes. Typically, crash reproduction is the first step in debugging software; however, manual crash reproduction can be very labour-intensive and time-consuming. The TU Delft group has applied a novel technique based on evolutionary algorithms to automatically replicate the reported crashes based on the crash stack traces. Future work will investigate the application of additional state-of-the-art methods to increase the range of software crashes supported by the automated technique.

Real insights

Another aspect of 3TU.BSR is data privacy, a huge concern when monitoring large software systems for things like medical devices. The project focuses on the sensitivity of the events collected from running software. Inald Lagendijk's TU Delft research group aims to adapt well-known anonymization and encryption methods for data publishing to software analysis and develop a privacy-preserving on-line conformance checking framework on event logs. As the anonymized or encrypted data might affect the accuracy of operations as well as add computational overhead, the framework should balance accuracy and computational performance with the level of protection and privacy provided.

To maintain the true mindset of in vivo analysis, it's important to use highly efficient techniques. The Twente group led by Jaco van der Pol focuses on developing parallel scalable algorithms for event analysis to support online recommendations. Multicore and symbolic model-checking techniques are applied to efficiently and correctly check for and predict abnormal behaviour.

Finally, visualization plays a crucial role during the whole project life cycle to make sure that the analysis results are easy to understand and provide real insights into system behaviour. The TUE research group led by Jack van Wijk is developing novel interactive visualization techniques for huge event streams generated by software execution. Appropriate graphical visualization for software behavioural patterns (or lack thereof) will help software analysts make initial hypotheses on how the software behaves and where anomalies might occur. To avoid scalability issues, the techniques developed should allow the analysts to specify what they are interested in, and show only a subset of the data using filtering, aggregation and abstraction.

Nour Assay (Eindhoven University of Technology), Vincent Bloemen (University of Twente), Mozhan Soltani (Delft University of Technology), Mariëlle Stoelinga (University of Twente) and Gamze Tillem (Delft University of Technology) are involved in the 3TU.BSR project.

Edited by Nieke Roos