3TU.BSR: Big Software on the Run

Summary

Millions of lines of code - written in different languages by different people at different times, and operating on a variety of platforms - drive the systems performing key processes in our society. The resulting software needs to evolve and can no longer be controlled a priori as is illustrated by a range of software problems. The 3TU.BSR research program will develop novel techniques and tools to analyze software systems in vivo - making it possible to visualize behavior, create models, check conformance, predict problems, and recommend corrective actions.

Prof.dr.ir. Wil van der Aalst (Eindhoven University of Technology)
Prof.dr. Arie van Deursen (Delft University of Technology)
Prof.dr. Jaco van de Pol (University of Twente)

Motivation: Big Software on the Run

Software forms an integral part of the most complex artifacts built by humans. Software systems may comprise hundreds of millions of program statements, written by thousands of different programmers, spanning several decades. Their complexity surpasses the comprehensive abilities of any single, individual human being. Accordingly, we have become totally dependent on complex software artifacts. Communication, production, distribution, healthcare, transportation, education, entertainment, government, and trade all increasingly rely on "Big Software". Unfortunately, we only recognize our dependency on software when it fails. Malfunctioning information systems of the Dutch police force and the Dutch tax authority, outages of electronic payment and banking systems, increasing downtime of high-tech systems, unusable phones after updates, failing railway systems, and tunnel closures due to software errors illustrate the importance of good software.

The following developments suggest that without a radical change of paradigm problems will only increase:

- *Growing complexity*: software systems do not operate in a stand-alone manner, but are increasingly inter-connected resulting in complex distributed systems.
- *Growing scale*: an increasing number of organizations use shared infrastructures (e.g. cloud computing), the number of devices connected to the internet is increasing, and an increasing amount of data is recorded (e.g. sensor data, RFID data, etc.).
- *Increasing diversity*: there is a growing diversity in platforms (covering traditional CPUs, multi-core architectures, cloud-based data centers, mobile devices, and the internet of things), versions (different releases having different capabilities), and configurations.
- *Continuous evolution of software*: late composition (components are assembled and connected while they are running), remote software updates (removing old errors but introducing new ones), and functional extensions (by merely changing running software) lead to unpredictable and unforeseen behavior.
- *Continuously changing environment*: the software must run in an ever changing context of (virtualized) hardware, operating systems, network protocols and standards, and must cope with wild variations of available resources, such as computer cores, bandwidth, memory and energy. Moreover, the software may be applied in ways not anticipated at design time.
- *Increasing demands related to security and trust*: as our reliance on software grows, concerns about security and privacy increase, while the software only becomes an ever more tempting target for attacks.

Taming the complexity of software has been an ongoing concern of computer science since its very inception. Traditionally, scientists attempt to ensure that software is built to satisfy stringent requirements. This a priori approach assumes that one has total control over the production process of all software and has demonstrated its value in stable environments. *However, the traditional a priori approach is unable to deal with the growing complexity and diversity of software systems operating in continuously evolving environments demanding on-the-fly changes to software.* Arguably, software systems are among the most complex artifacts humanity has ever produced. However, we expect a software system to run on different platforms, cooperate with an array of unknown systems, be used in a way not envisioned at design time, and adapt to changing requirements. To meet these high expectations, we need to really understand complex evolving software systems and consider this one of the grand challenges of our lifetime.

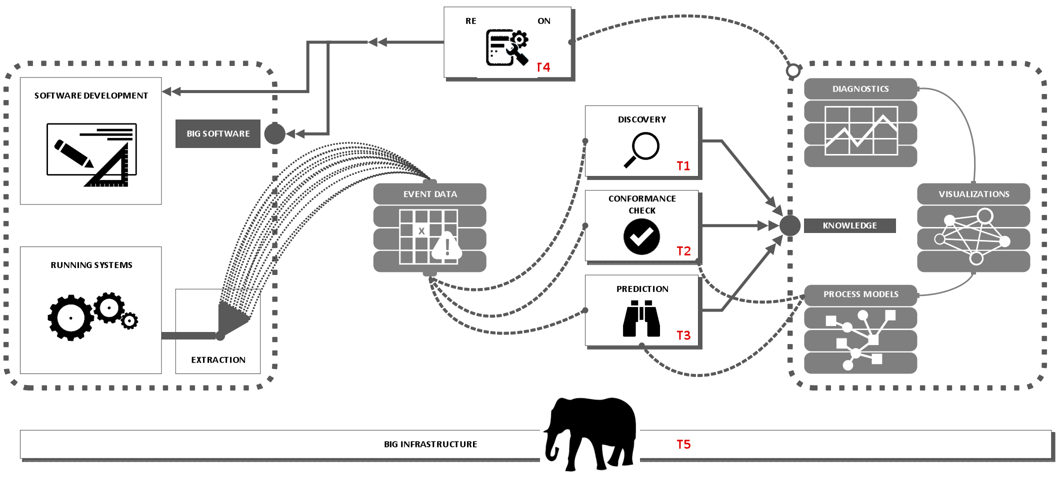
To deal with *Big Software on the Run (BSR)*, we propose to *shift the main focus from a priori software design to a posteriori software analytics* thereby exploiting the large amounts of event data generated by today's systems. The core idea is to study software systems *in vivo*, i.e., at runtime and in their natural habitat. We would like to understand the actual (desired or undesired) behavior of software. Running software needs to adapt to evolving and diverging environments and requirements. This forces us to consider software artifacts as "living organisms operating in changing ecosystem". *This paradigm shift requires new forms of empirical investigation that go far beyond the common practice of collecting error messages and providing software updates.*

Hence, there is an urgent need to develop innovative techniques to discover how systems really function, check where systems deviate from the desired and expected behavior, predict the reliability, performance and security over time, and recommend changes to address current or future problems. These techniques need to be able to deal with torrents of event data ("Big Data") and extremely complex software ("Big Software"). The urgency of software-related problems and the opportunities provided by such a new focus justify a dedicated BSR research program.

3TU.BSR: NIRICT's Response to Big Software on the Run

The goal of the highly innovative 3TU.BSR research program is to provide a solid scientific basis for in vivo software analytics while exploiting the world-renowned 3TU computer science groups working on process mining, visualization, software engineering, formal methods, security analysis, and distributed/large-scale computing. Due to his ground-breaking work on the workflow patterns, workflow verification, and process-aware information systems, Van der Aalst is widely recognized as the leading business process management researcher in the world. He was the first to see the value of process mining and developed various process discovery and conformance checking techniques. Van Deursen pioneered software engineering's first steps into automated testing of modern web applications. He also investigated the interplay between coding activities and developer testing activities. Van de Pol is a well-known expert on high-performance analysis of discrete systems and developed many parallel algorithms for state space generation, reduction, model checking, and cycle analysis. Next to these three representatives, many other NIRICT researchers are involved, e.g., Van Wijk working on visualization, Lagendijk working on statistical and information-theoretical methods for pattern recognition, Zaidman working on the analysis of software repositories, Huisman working on runtime conformance checking, and Katoen working on stochastic processes.

Within 3TU.BSR we have identified five tracks. These tracks cover the different areas where major breakthroughs are needed.



In the *discovery track* (Track **T1**) the primary focus is on creating more abstract representations of the massive amounts of event data. We will develop techniques for generating models and visualizations showing what is really going on in a software system or collection of systems. This analysis is of an explorative nature, i.e., there is no clear understanding of the problem. Based on the discovery track we will be able to determine relevant symptoms used in other tracks. In the *conformance checking track* (Track **T2**) we will develop techniques to detect deviations from some normative or descriptive model (possibly a model discovered in Track **T1**). The model may be based on domain knowledge or based on the symptoms identified in the discovery track. In the *prediction track* (Track **T3**) we will develop techniques to predict functional and non-functional properties of running cases, individual systems, and classes of systems. Whereas the discovery and conformance checking tracks (**T1** and **T2**) only consider past behavior, this third track will aim to predict behavior (building on the results of **T1** and **T2**). In the *recommendation track* (Track T4) we translate the results of the first three tracks (**T1**, **T2**, and **T3**) into recommendations related to the software system or the development process. Recommendations may provide input for testing, debugging, reuse, reconfiguration, and adaptation. In the *big infrastructure track* (Track **T5**) we focus on distributed and large-scale computing. The main goal of this track is to enable discovery, conformance checking, predictions, and recommendations at the unprecedented scale BSR aims at. This includes the extraction and storage of event data, privacy concerns, interoperability, and massive parallelization. The infrastructure should also be able to work in conjunction with modern technologies (clouds, grids, mobile devices, etc.).

We have identified six subtracks focusing on the most pressing challenges in **T1-T5**. We also identified common case studies (e.g., the Open Source Eclipse Ecosystem) to ensure collaboration between all subtracks.

3TU.BSR: Organization

The management team of 3TU.BSR consists of Prof.dr.ir. Wil van der Aalst (Eindhoven University of Technology), Prof.dr. Arie van Deursen (Delft University of Technology), Prof.dr. Jaco van de Pol (University of Twente). Van der Aalst chairs the management team and serves a scientific director.

The following chairs/groups are involved:

- The *Architecture of Information Systems* (AIS) group at *Eindhoven University of Technology* (Van der Aalst).
- The *Visualization* (VIS) group at *Eindhoven University of Technology* (Van Wijk).
- The *Software Engineering Research Group* (SERG) at *Delft University of Technology* (Van Deursen)
- The *Cybersecurity Group* (CY) at *Delft University of Technology* (Lagendijk)
- The *Formal Methods and Tools* (FMT) at *University of Twente* (Van de Pol & Huisman)

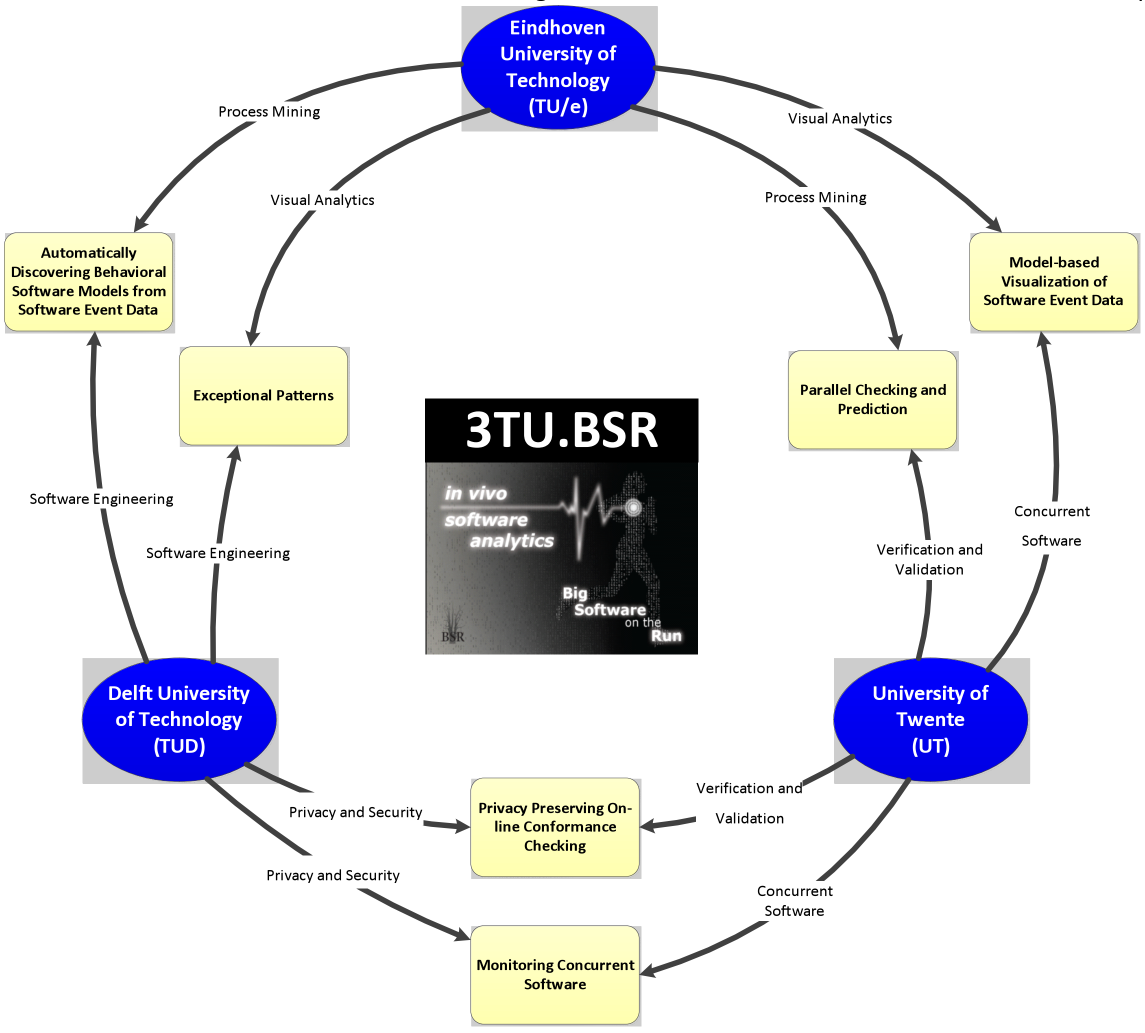
Nine researchers will be appointed: 6 PhDs and 3 postdocs. In order to warrant sustainable embedding of the research in 3TU.Nirict and continuity after the end of the funded period, each PhD will be supervised by tenure track and senior faculty from two different universities. Note that, next to the group leaders, also tenure track researchers will be actively involved in the program.

3TU.BSR: Competences and Collaboration

Within 3TU.BSR we join the following six competences:

1. **Process Mining** (Van der Aalst, Van Dongen, Fahland). The AIS group is well known for its research in process analytics. Traditionally there is a gap between model-based analysis and more data-driven approaches (data mining, machine learning, business intelligence and evolutionary algorithms). Process mining techniques aim to bridge this gap. The expertise of AIS in process model discovery, conformance checking, data-driven model repair and extension, bottleneck analysis, and prediction is used to analyze software based on the events recorded.
2. **Visual Analytics** (Van Wijk, Van de Wetering, Westenberg). The VIS group has a strong track record in information visualization and visual analytics, especially for applications in software engineering. The expertise on the development of novel methods, techniques, and systems for interactive visualization and analysis is used to enable experts to find patterns, trends, and form hypotheses about huge streams of event data.
3. **Software Engineering** (Van Deursen, Zaidman, Bacchelli). The SERG has a strong reputation for its research in software testing, software architecture, repository mining, collaborative software development, and the use of dynamic analysis for the purpose of system understanding. This software engineering expertise is of crucial importance for the analysis of running software based on data.
4. **Privacy and Security** (Lagendijk, Erkin, van der Lubbe). The CY group concentrates its research on the intersecting field of big data and privacy enhancing technologies. Solutions developed for preserving privacy of data by anonymizing methods and in particular by processing data under encryption, will be used for finding patterns, trends, and analysis of sensitive data obtained from running software. A new challenge in BSR is the use of privacy enhancing techniques in on-line assimilating and conformance testing using event data and quantitative software models.
5. **Verification and Validation** (Van de Pol, Stoelinga, Langerak, Katoen). The FMT group combines a unique expertise in scalable model checking algorithms (van de Pol, Katoen), quantitative modeling and evaluation (Katoen, Stoelinga, Langerak), and model based testing (Stoelinga, vd Pol). We will explore the scope of model based testing and redesign its algorithms for online conformance checking of big software in Track 2. Scalable model checking algorithms that combine multi-core and symbolic techniques, will be developed to predict the behavior of runtime systems in Track 3.
6. **Concurrent Software** (Huisman, Rensink, Aksit). Marieke Huisman (winner of the Dutch ICT award 2013) and her team have a strong track record on analysis of concurrent software. In her ERC project VerCors, she developed tools for static analysis of concurrent programs, using different concurrency paradigms (shared memory, but also vector programs). Within BSR, these analysis techniques will be extended to a dynamic setting, to obtain quick feedback, prevent serious problems, and to create structured data to facilitate data mining. To move towards this dynamic setting, ingredients from runtime monitoring (Aksit) and software transformation (Rensink) will be incorporated.

The goal of 3TU.BSR is to stimulate collaboration. The figure below shows the collaboration the the 6 PhD projects.


Eindhoven University of Technology (TU/e)
Process Mining
Visual Analytics
Visual Analytics
Process Mining
Automatically Discovering Behavioral Software Models from Software Event Data
Exceptional Patterns
Parallel Checking and Prediction
Model-based Visualization of Software Event Data
3TU.BSR
in vivo software analytics
Big Software on the Run
Software Engineering
Software Engineering
Verification and Validation
Concurrent Software
Delft University of Technology (TUD)
University of Twente (UT)
Privacy and Security
Privacy Preserving On-line Conformance Checking
Verification and Validation
Privacy and Security
Concurrent Software
Monitoring Concurrent Software

3TU.BSR: Push for Open-Source Data and Software

Given the topic, the IT infrastructure is crucial. Each of the participating universities already has an infrastructure (both hard and software). 100K€ has been reserved for aligning these infrastructures and modest investments (e.g., special software or storage). We plan to use the limited resources creatively, e.g., hire excellent students as assistants. 50K€ has been reserved for various meetings.

Trace data will be shared among the different groups through the so-called **BSR-Repository** and serve as an important catalyst for collaboration. The traces will come from a variety of sources. Initially the data will come from systems that are currently being studied by the various groups. For example, the TU/e process mining group has collected traces from dozens of organizations (both industrial and open-source). Another example is the Failure Trace Archive, in which execution traces with failure information are collected for various (distributed) systems. Moreover, in the joint case studies new sources of event data will be collected. Existing and new event data will be made available through the BSR- repository and will serve as an important catalyst for collaboration.

Next to the **BSR-Repository** (for storing and sharing event data and models), we will develop two key tool sets: the **BSR-Extractor** (for extracting event data from a variety of systems) and the **BSR-Analyzer** (for all analysis techniques developed). This will be a collaborative effort leveraging existing software (e.g., ProM). The **BSR-Analyzer** will be one of the key results of 3TU.BSR and provide dozens of powerful analysis techniques. Such a toolset could not be realized without a substantial investment, for which we request the support of 3TU.

3TU.BSR: Summary

3TU.BSR joins a unique set of competences that are highly complementary. The 3TU.BSR research program will develop novel techniques and tools to analyze software systems in vivo - making it possible to visualize behavior, create models, check conformance, predict problems, and recommend corrective actions. In the design of the program we made sure that collaboration is stimulated and that results are consolidated in terms of shared data and software. Moreover, faculty members (e.g., tenure track researchers) of different universities will be involved in joint PhD projects. We will also seek funding to extend the scope and increase volume (e.g., through the NWO/EW topsector program on Big Software. These efforts ensure that we maximize the impact of 3TU.BSR in the short term while enabling collaboration that extends far after the initial four year period.

Appendix

Automatically Discovering Behavioral Software Models from Software Event Data (part of Track T1) Van der Aalst & Van Deursen

Process models and user interface workflows underlie the functional specification of almost every substantial software system. However, these are often left implicit or are not kept consistent with the actual software development. When the system is utilized, user interaction with the system can be recorded in event logs. After applying process mining methods to logs, we can derive process and user interface workflow models. These models provide insights regarding the real usage of the software and can enable usability improvements and software redesign. In this project, we aim to develop process discovery techniques specific for software. How can domain knowledge and software structure be exploited while mining? How to discover software patterns and anti-patterns?

Model-based Visualization of Software Event Data (part of Track T1) Van Wijk & Huisman

Visualization can be a powerful means for understanding large and complex data sets, such as the huge event streams produced by running software systems. During explorative analysis (T1) experts have to be enabled to see what patterns occur, during monitoring (T2) anomalous events and patterns have to be detected, where in both cases we can exploit the unique capabilities of the human visual system. However, simply showing events as a sequence of items will fall short because of lack of scalability. The challenge is to enable users to specify what they are interested in, and to show only a limited subset of the data, using filtering, aggregation, and abstraction. We propose to enable users to define models for this, ranging from simple range filters to process models. We will study which (combinations of) models are most appropriate here, such that occurrences of events, temporal and logical patterns, and the relations between occurrences and attributes of events can be detected, and to facilitate analysts to define and check hypotheses on patterns.

Exceptional Patterns (part of Tracks T1 and T4) Van Deursen & Van Wijk

A particularly challenging phenomenon in software development are 'exceptions'. Most programming is focused on 'good weather behavior', in which the system works under normal circumstances. Actual deployment however, often takes place in a changing or unexpected environment. This may lead to exceptions being raised by the application, which should be handled by the application. Unfortunately, predicting such exceptional circumstances is often impossible. Consequently, developers have difficulty adequately handling such exceptions. Some exceptions are simply swallowed by the applications, others are properly logged, and yet other may lead to unpredictable behavior. To resolve this, we propose to analyze log files for 'exceptional patterns' -- patterns that hint at the presence of exceptions. To find such patterns, we propose to use visualization techniques applied to log data and stack traces. Furthermore, we will investigate ways to predict future occurrences of exceptions, and recommendations on how to improve exception handling in the code base.

Monitoring Concurrent Software (part of Tracks T1 and T2) Huisman & Lagendijk

The goal is to develop a monitoring system for concurrent software. Making monitoring transparent is the big challenge: monitoring should not affect program behavior. A general purpose approach will be designed, based on local annotations and global properties. Runtime monitoring is essential to check conformance of concurrent software during deployment (T2). At the same time, runtime monitoring provides insight in low-level software events, generating a continuous data stream of events that feeds discovery (T1). With process mining and visualization technology in Eindhoven, we will explore the scope of concurrent software monitoring.

Privacy Preserving On-line Conformance Checking (part of Track T2) Lagendijk & Van de Pol

Privacy enhancing techniques have been applied dominantly to data analysis problems (such as pattern recognition) and multimedia algorithms (such as recommendation engines). The goal of privacy preserving on-line conformance checking is to research the problem of privacy and security protection in software engineering for the first time. The central problem is that conformance checking algorithms may need to operate on event data that is sensitive in some way, for instance, contains user-related information. Such data can be anonymized or encrypted for protection, yet this might affect the accuracy of the conformance checking procedure. It will therefore be necessary to find an acceptable trade-off between the level of protection, the utility of the results obtained from the privacy-enhanced version of the conformance checking algorithm, and the additional computational overhead introduced by the anonymization or encryption process.

Parallel Checking and Prediction (part of Tracks T2 and T3) Van de Pol & Van der Aalst

Based on the models discovered by online observations (Track 1), the goal of this research project is to develop scalable technology for predicting future system behavior (Track 3). Assuming that the system’s components will behave similar to the process models learnt so far, (quantitative) model checking techniques will be applied to explore possible runs and interactions of the integrated system. In order to support online recommendations (Track 4), the model checking results should be available nearly instantaneously. This calls for parallel, scalable algorithms that will be run on BSR.cloud infrastructure in Twente. Large scale distributed experiments will be run on DAS5.