behavioural properties, resource-management policies and self-adaptation strategies can be clearly specify and implement component coordination on a high-level of abstraction, in such a manner that one hand, and adaptation policies, on the hand. These models will form the foundation of a rigorous Theoretical work within the project will aim to develop expressive formal models for the specification with information about resource availability to optimise overall system performance.

The main goal of this post-doctoral project is to extend JavaBIP with new mechanisms for resource management and self-adaptation. Indeed, the environment of modern systems is inherently highly variable. In particular, this is due to interferences among applications sharing common resources and to the migration, e.g. of cloud applications among computing units. Thus, fluctuations of resource availability become the norm rather than an exception. Instead of waiting for the resources to become available, applications adapt their behaviour to the changes in the environment. Mechanisms representing resource availability and dependencies must be explicitly provided in the design framework. To enable efficient coordination, components must advertise their resource requirements to the coordinating engines, which in turn must be able to combine such requirements with information about resource availability to optimise overall system performance.

Assignments

When building large concurrent systems, one of the key difficulties lies in coordinating component behaviour and, in particular, management of the access to shared resources of the execution platform. A simple example consists in managing the memory usage by a set of concurrent components, such as Camel routes [1]. A Camel route connects a number of data sources to transfer data among them. The data can be fairly large and may require additional processing. Hence, Camel routes share and compete for memory. Without additional coordination, simultaneous execution of several Camel routes can lead to OutOfMemory exceptions, even when each route has been tested and sized appropriately on its own. Since, in concurrent environments, it is practically infeasible to envision all possible execution scenarios, synchronization errors can result in race conditions and deadlocks.

To address this concurrency challenge, we have developed JavaBIP [2]. JavaBIP is a Java adaptation of the Behavior, Interaction, and Priority (BIP) framework [3], providing two primitive mechanisms for component coordination: (i) multi-party synchronization of component transitions and (ii) asynchronous event notifications.

The main goal of this post-doctoral project is to extend JavaBIP with new mechanisms for resource management and self-adaptation. Indeed, the environment of modern systems is inherently highly variable. In particular, this is due to interferences among applications sharing common resources and to the migration, e.g. of cloud applications among computing units. Thus, fluctuations of resource availability become the norm rather than an exception. Instead of waiting for the resources to become available, applications adapt their behaviour to the changes in the environment. Mechanisms representing resource availability and dependencies must be explicitly provided in the design framework. To enable efficient coordination, components must advertise their resource requirements to the coordinating engines, which in turn must be able to combine such requirements with information about resource availability to optimise overall system performance.

Theoretical work within the project will aim to develop expressive formal models for the specification and analysis of platform capacities and application requirements for various kinds of resources, on one hand, and adaptation policies, on the hand. These models will form the foundation of a rigorous design approach—to be implemented in JavaBIP—which would provide resource management and self-adaptation mechanisms along with appropriate specification languages, allowing designers to specify and implement component coordination on a high-level of abstraction, in such a manner that behavioural properties, resource-management policies and self-adaptation strategies can be clearly specified and managed in a declarative manner. In particular, this is due to interferences among applications sharing common resources and to the migration, e.g. of cloud applications among computing units. Thus, fluctuations of resource availability become the norm rather than an exception. Instead of waiting for the resources to become available, applications adapt their behaviour to the changes in the environment. Mechanisms representing resource availability and dependencies must be explicitly provided in the design framework. To enable efficient coordination, components must advertise their resource requirements to the coordinating engines, which in turn must be able to combine such requirements with information about resource availability to optimise overall system performance.

Context

The Spirals project-team is conducting research activities in the domains of distributed systems and software engineering. Spirals aims at introducing more automation in the adaptation mechanisms of software systems, in particular, transitioning from adaptive systems to self-adaptive systems. For that, we investigate solutions from several disciplines, such as formal methods, data mining, machine learning, and distributed algorithms. This contributes to the goal of obtaining eternal distributed systems. More information can be found on our website https://team.inria.fr/spirals/.

Assignment

When building large concurrent systems, one of the key difficulties lies in coordinating component behaviour and, in particular, management of the access to shared resources of the execution platform. A simple example consists in managing the memory usage by a set of concurrent components, such as Camel routes [1]. A Camel route connects a number of data sources to transfer data among them. The data can be fairly large and may require additional processing. Hence, Camel routes share and compete for memory. Without additional coordination, simultaneous execution of several Camel routes can lead to OutOfMemory exceptions, even when each route has been tested and sized appropriately on its own. Since, in concurrent environments, it is practically infeasible to envision all possible execution scenarios, synchronization errors can result in race conditions and deadlocks.

To address this concurrency challenge, we have developed JavaBIP [2]. JavaBIP is a Java adaptation of the Behavior, Interaction, and Priority (BIP) framework [3], providing two primitive mechanisms for component coordination: (i) multi-party synchronization of component transitions and (ii) asynchronous event notifications.

The main goal of this post-doctoral project is to extend JavaBIP with new mechanisms for resource management and self-adaptation. Indeed, the environment of modern systems is inherently highly variable. In particular, this is due to interferences among applications sharing common resources and to the migration, e.g. of cloud applications among computing units. Thus, fluctuations of resource availability become the norm rather than an exception. Instead of waiting for the resources to become available, applications adapt their behaviour to the changes in the environment. Mechanisms representing resource availability and dependencies must be explicitly provided in the design framework. To enable efficient coordination, components must advertise their resource requirements to the coordinating engines, which in turn must be able to combine such requirements with information about resource availability to optimise overall system performance.

Theoretical work within the project will aim to develop expressive formal models for the specification and analysis of platform capacities and application requirements for various kinds of resources, on one hand, and adaptation policies, on the hand. These models will form the foundation of a rigorous design approach—to be implemented in JavaBIP—which would provide resource management and self-adaptation mechanisms along with appropriate specification languages, allowing designers to specify and implement component coordination on a high-level of abstraction, in such a manner that behavioural properties, resource-management policies and self-adaptation strategies can be clearly specified and managed in a declarative manner. In particular, this is due to interferences among applications sharing common resources and to the migration, e.g. of cloud applications among computing units. Thus, fluctuations of resource availability become the norm rather than an exception. Instead of waiting for the resources to become available, applications adapt their behaviour to the changes in the environment. Mechanisms representing resource availability and dependencies must be explicitly provided in the design framework. To enable efficient coordination, components must advertise their resource requirements to the coordinating engines, which in turn must be able to combine such requirements with information about resource availability to optimise overall system performance.
stated, combined and enforced.

References


Main activities

Main activities:

- Scientific research (definition of models, algorithms etc.; proofs)
- Implementation of prototype tools for evaluation of the proposed techniques
- Written presentation of the obtained results through papers and reports
- Oral presentation of the obtained results at scientific conferences
- Participation in the supervision of students at all levels

Additional activities:

- Strengthening of one’s scientific network and definition of a career strategy
- Participation in the development of a user community
- Participation in other activities to promote the team’s research to broader audiences

Skills

- Formal methods (in particular semantic models, e.g. finite automata, Labelled Transition Systems and Petri Nets; behavioural equivalences, e.g. trace equivalence and bisimilarity)
- Verification (in particular temporal logics, e.g. LTL and CTL; tools, e.g. nuXmv, mCRL2)
- Knowledge of coordination languages, such as BIP, is a plus
- Proven experience in preparation of scientific documents (including mastery of LaTeX)
- Proven experience in software development (Java, Python)
- Excellent communication skills

Benefits package

- Subsidised catering service
- Partially-reimbursed public transport

Remuneration

Around 31 000 € yearly brutto.