A new generation of software systems (for example smart health support, building Challenge 2. Joint and coordinated
RQ1. How to reduce the development cost of usable and highly interactive DSL front-specific undo algorithms, complex user interactions. To summarize we defined the that exhibits critical flaws for supporting modern interactive features, such as domain-
recent work [4] that the current UI toolkits still rely on the UI event processing model requires further research work on how to program user interactions: we shown in our interactive features must go beyond the classical use of mice, keyboards, and standard widgets (buttons, menus) to leverage the advances made by the HCI community. This
want to bring those features to another DSL we have to re-code them. Researchers need
modern navigation features on one DSL [3]. We coded those features by hand. If we adoption of DSLs. For example, in our own research work we studied the impact of
domain-specific undo algorithms, complex user interactions. Moreover, Such features [1]. Current IDEs hardly support such collective intelligence as it requires novel features

tabular). Experts use numerous DSLs broadly used in the industry, sometimes without knowing that they handle a DSL. For example, conceiving an air plane usually involves electricity, resistance, scientific computing schema/models, that follow a specific nomenclature (i.e. the DSL grammar), with a specific representation (i.e. the DSL concrete syntax), with editing environments. Similarly, for conceiving a Web application experts may handle Docker, Git-Lab CI (continuous integration), ANTLR (grammar definition), CSS (style definition), Kubernetes (container orchestration) models/configuration files. All these languages are DSLs that experts use for completing a specific task. If a DSL works by itself, its use may impact other DSL models. For example, an electrician engineer that apply changes on his DSL models may have impacts on DSL models of other experts, such as on heating models. Each of those experts brings their own viewpoint, based on their expertise, for solving the global challenge, developing the system. This phenomenon is known as collective intelligence [3]. Current IDEs hardly support such collective intelligence as it requires novel features to share, communicate, explore, synchronize experts works. Moreover, Such features and the way they are used within a IDE depend on the experts domain. Language engineers, that develop IDEs specifically for DSLs, thus struggle in coding by hand such complex features.

Assignment

Objectives

Challenge 1. Proper DSL Usability

On one side, researchers focus on improving the creation and maintenance process of DSL back-ends to reduce their development cost. On the other side, researchers study how to improve the usability, the interactivity of some DSL front-ends to match new usages. Few research work, however, focus on new techniques for reducing the cost of building usable DSL front-ends [2]. So, language engineers still have to customize the interactivity of DSL environments by hand. Such tasks are costly and may hinder the adoption of DSLs. For example, in our own research work we studied the impact of modern navigation features on one DSL [3]. We coded those features by hand. If we want to bring those features to another DSL we have to re-code them. Researchers need to include interactivity within the creation and maintenance process of DSLs. Those interactive features must go beyond the classical use of mice, keyboards, and standard widgets (buttons, menus) to leverage the advances made by the HCI community. This requires further research work on how to program user interactions: we shown in our recent work [4] that the current UI toolkits still rely on the UI event processing model that exhibits critical flaws for supporting modern interactive features, such as domain-
specific undo algorithms, complex user interactions. To summarize we defined the following research question to address:

RQ1. How to reduce the development cost of usable and highly interactive DSL front-ends?

Challenge 2. Joint and coordinated DSLs usability

A new generation of software systems (for example smart health support, building energy management, or intelligent transportation systems) gives new opportunities for leveraging DSLs. The development of those systems requires expertise in multiple
domains. So, different types of experts (e.g. scientists, engineers from different domains) must work in a coordinated manner on various aspects of the system across multiple development phases. Those experts can use DSLs to support their work that focuses on a specific system aspect. The support and integration of DSLs lead to what researchers called the globalization of modeling languages [5], i.e. the use of multiple languages for the coordinated development of diverse aspects of a system. So, while the previous point focuses on the proper interactivity of a DSL, i.e. when a DSL is used alone, in various cases experts use multiple DSLs conjointly. This is what we call the joint and coordinated use of DSLs: such a use of multiple DSLs can lead to a tower of Babel [6] (i.e. a cacophony of DSLs) when the coordination of experts and their DSL models is not achieved correctly. For example the development of air-plane engines involves various experts from different domains: an electrician engineer that apply changes on dedicated DSL models may have impacts on models of an heating engineer; a change in CAD models may impacts material resistance models; when physicians apply changes on their physical models, this may impact system engineers' models. All those models, made using various DSLs (e.g. Modelica, Simulink) are thus dependent each others. This makes the development process more complex: how domain experts can understand changes that affect their models if those changes come from models out-of-the-scope of their expertise? How experts from different domains can easily iterate on changes to converge to an optimal solution? We consider this problem as a usability issue that affects both development environments and processes, to lead to the following research question.

RQ2. How to improve the usability of using several DSLs conjointly or in coordination?

Main activities
The candidate will work on these two scientific challenges to produce both high level scientific contributions published in top venues, and develop prototypes to assess the contributions.

Skills
Application conditions
We are looking for exceptional and motivated candidates for this 3-year PhD. The candidate must have (or be about to obtain) a master’s or engineering degree in computer science. A mastery of scientific English is necessary. Knowledge of French is not required. Gross monthly salary: around 2050 € (years 1 and 2) then around 2150 €.

Environment
The candidate will work in the DiverSE team, common to IRISA and Inria. The DiverSE team is located in Rennes. DiverSE's research is in the field of software engineering. The team is actively involved in European, French and industrial projects and is composed of 9 professors/researchers, 20 PhD students, 4 post-docs and 3 engineers. The main supervisors of the thesis will be Arnaud Blouin, and Benoit Combemale. The candidate will enroll in the doctoral school in computer science of the University of Rennes 1.

Benefits package
- Subsidized meals
- Partial reimbursement of public transport costs
- Possibility of teleworking (90 days per year) and flexible organization of working hours
- Partial payment of insurance costs

Remuneration
Monthly Gross salary:
- 2051 euros for the 1st two years
- 2158 euros on 3rd year

Recruitment Policy:
As part of its diversity policy, all Inria positions are accessible to people with disabilities.