Lero is the Irish Research Centre for Software Engineering. Lero provides an interactive and international environment for software engineering research inspired by concrete industry cases.

We have several interesting topics for MSc theses or internships. To give you an overview, some first information is described below. More concrete task descriptions will be defined individually in coordination with the students before their stay at Lero.

**CONTACT**

For questions and for applications please contact Andreas Pleuss, andreas.pleuss@lero.ie

---

**AREA 1: FEATURE-ORIENTED SOFTWARE EVOLUTION**

Software is continuously evolving [1], e.g., to cope with new customer requirements, changing markets, and technological progress. Keeping track and making the right decisions about long-term evolution of a system is a major challenge, especially for large and complex systems. For instance, it needs to be decided about the most important changes to be implemented until the next release (e.g., new features) while taking into account the impact of changes (e.g., some features might require difficult code refactorings).

Bridging the gap between the high-level business strategy and technical low-level details can be difficult. The concept of **features** [2] provides a good granularity to bridge this gap. Features are usually comprehensible for non-technical stakeholders (e.g., customers, managers) but they can also be related to concrete code artifacts [3]. Systematic evolution planning and monitoring on feature level [4] can help to manage the challenges of software evolution.

**Evolution planning** includes techniques for decision making, e.g., collecting new candidate features and prioritizing them. **Evolution monitoring** includes data extraction (e.g., from source code repositories and customer feedback) and its aggregation. Planning and monitoring should both be supported by appropriate visual tool support such as a “dashboard” for feature-oriented evolution.

Theses in this area will contribute to the development of concepts and tool prototypes for feature-oriented evolution planning and monitoring, e.g., as plug-ins for the Eclipse IDE [5].

---

**AREA 2: INTERACTIVE TOOL SUPPORT FOR PRODUCT CONFIGURATION**

Today many software products are highly configurable to serve individual customer needs while reusing the same base product. For instance, **software products lines** [6][7] enable to develop a whole family of products from a set of shared software assets. The variability between the different products in a product line is defined, e.g., in a **feature model** [2] which specifies the available features and the dependencies between them. A concrete product is then specified by a **product configuration**, i.e., selecting the features that should be included in the product (while considering the dependencies between features).

In practice, feature models become very large and complex with hundreds or thousands of features. Visual tool support (e.g., our **S2T2 Configurator** [8]) can help to reduce the cognitive complexity for engineers when configuring a product. The tool includes a reasoning engine which ensures that a configuration adheres to the dependencies defined in the feature model and supports the user in resolving conflicts.

Theses in this area will develop novel concepts to support product configuration and implement them as part of the S2T2 Configurator tool. This can include novel visualizations, e.g., to support trade-off analysis (e.g., performance vs. costs of a product), extending the reasoning engine (e.g., to support numerical attributes, such as costs and performance of a feature), and performing user tests. S2T2 is implemented in Java, using the Eclipse Modeling Framework [9], SAT4J [10], and Prefuse [11].
Software Product Lines (see Area 2) enable efficient derivation of products from a set of shared software assets. For instance, model-driven techniques [12] enable to automatically generate a product according to a given product configuration. However, automatically generated products are not always sufficient in practice, in particular with respect to non-functional properties such as usability or performance. For instance, generated user interfaces do not provide sufficient quality required for professional products.

Hence, there is a need to customize generated products. However, just changing the generated code manually is not useful as these changes would be lost whenever an updated version of the product is generated. Instead, there needs to be a tool-supported way to specify customizations so that they can be stored and reused. For instance, there are existing domain-specific modeling languages that support modeling a user interface [13]. Such models can be used to specify user interface customizations and store them for reuse in future product versions and in other products that require the same customizations [14].

Theses in this area will investigate model-driven concepts and develop tool prototypes to support developers in customizing generated products, e.g., on the user interface level. Tools should allow to store, manage and reuse customizations. There is also a need for interactive, visual tools that can be used by developers who are not experts in modeling, such as user interface designers. Model-driven technologies will be implemented using the frameworks from the Eclipse Modeling Project [15].

LITERATURE