EXEMPLAR: An Experimental Information Repository for Software Engineering Research

Jose Antonio Parejo, Sergio Segura, Pablo Fernandez, and Antonio Ruiz Cort´es

University of Sevilla, Spain
ETSII. Avda. de la Reina Mercedes s/n, 41012, Sevilla. japarejo@us.es

Abstract. The number and variety of experiments carried out in software engineering research is growing, leading to an increasing need of replication and review. In order to support such needs the information about experiments should be provided as lab-packs. However, this information is often scattered, poorly structured, and even unavailable, implying a tedious process of search and gathering. EXEMPLAR is an online platform for managing experimental information, which allows the uploading and publication of experimental lab packs, and an efficient search. The platform also supports the use of formal languages for providing experimental descriptions (e.g. SEDL and MOEDL). In so doing, EXEMPLAR enables the automated analysis of lab-packs, in order to detect common validity threats and missing information which could hinder replicability.

Keywords: empirical research, experiments, experimental replicability, experimental repositories

1 Introduction

In science, the quality of an experiment is determined primarily by two factors: its degree of validity and its replicability. According to [1], "*The use of precise, repeatable experiments is the hallmark of a mature scientific or engineering discipline*". In order to achieve replicability and enable validity checking, the information about experiments should be provided as lab-packs comprising of: a description of the experiment, the materials used and data generated during the conduction, and the results of the analyses performed on such data. When search based techniques are used in the experiments, providing a comprehensive experimental description becomes even more difficult, given their high number of parameters and stochastic nature.

EXEMPLAR (EXpErimets Management PLAtfoRm) is an online repository of experimental information that aims at supporting the creation of high quality experiments. EXEMPLAR focuses on easing the publication of lab-packs, supporting efficient search, and assuring the quality of experimental information through the automated detection of common validity threats and missing information (based on the analysis of experimental descriptions when provided in

formal languages such as SEDL [2] and on the analysis of lab-packs contents). The repository is available for public access at https://exemplar.us.es.

The remainder of this paper is structured as follows: section 2 describes the main features of EXEMPLAR, including the support for authoring experimental descriptions in two DSL defined by the authors in [2], SEDL and MOEDL. Section 3 succinctly describes the advantages of using a model driven transformation between documents in such languages. Finally, section 4 provides some conclusions and describes future work.

2   EXEMPLAR features

2.1   Experimental information repository

EXEMPLAR is essentially an online repository of experimental information that provides: i) support for uploading and controlling the availability of experimental information, through the creation of workspaces that can contain several labpacks; ii) support for the creation of succinct, precise and unambiguous description of experiments, by aiding the authoring documents written in languages created specifically for that purpose; and iii) support for searching experiments based on keywords, several classification taxonomies, and on the indexation of the contents of the labpacks. Those features are described in detail below. **Information Organization: Workspaces, labpacks and access control**. In EXEMPLAR, each registered user has its own personal space with a maximum quota (currently limited to 1Gb). In such personal space, users can create an unlimited number of workspaces, for which they can control access.
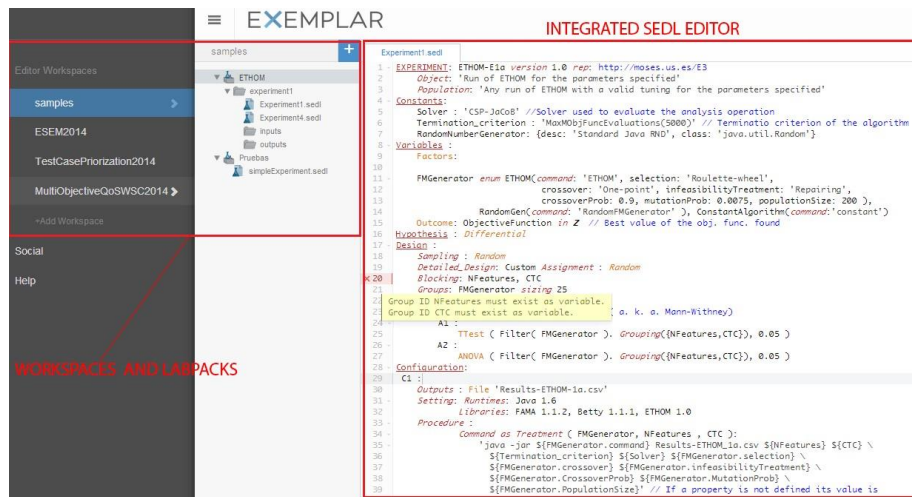

Fig.1. EXEMPLAR interface for workspace and lab-packs management

Currently workspaces are either public (meaning availability to anyone in read-only mode) or private, but authors plan to implement workspace sharing among users in read-write mode as future work. Each workspace contains an unlimited number of lab-packs. Each lab-pack contains the information of a single experiment, structured as an unlimited number of files and nested folders. The workspaces and labpacks management interface of EXEMPLAR is shown in figure 1 (left side). **Search and Indexation**. EXEMPLAR supports two different search mechanisms. On the one hand, users can search labpacks based on their tags or classification according to standard taxonomies (for software engineering experiments we support the SWEBOK taxonomy of areas [1]). On the other hand, users can perform full text searches on the indexed contents of the labpacks. Figure 2 shows the search page of EXEMPLAR.
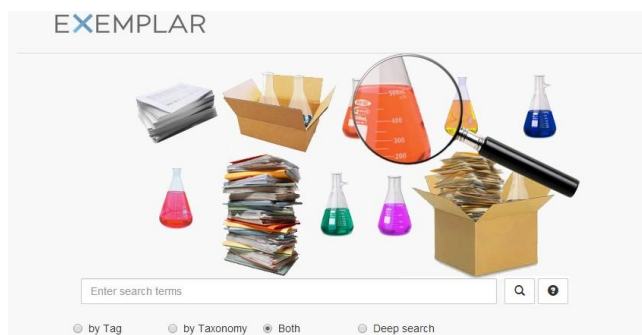
Fig.2. Experimental information search in EXEMPLAR

2.2 Support for experimental description and lab-packs layout

Apart from its capabilities as information repository, EXEMPLAR supports the creation of formal descriptions of experiments based on SEDL [2], and it provides a default lab-pack directory layout for inducing a tidy structure on lab-packs contents.
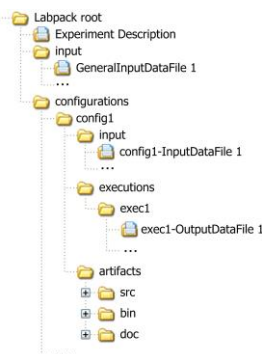**Formal experimental descriptions with SEDL and MOEDL**. Formal description of experimental information in EXEMPLAR is supported through integrated SEDL and MOEDL editors[1]. **SEDL** (Scientific Experiment Description Language) is a generic language to describe experiments in a precise, unambiguous and tool-independent way. SEDL documents include the information that any experimental

---

[1] The SEDL editor of EXEMPLAR is complete and supports the whole syntax described in [2]. The MOEDL editor of EXEMPLAR is currently under development, and it is not as usable.

description should provide regardless of the application domain: objects, subjects, population, variables, hypothesis, treatments and analysis to be performed. A detailed description of the syntax of SEDL along with several examples are provided in [2]. Figure 1 shows that integrated editor support **syntax colouring**, **sections code folding**, **auto-save** and **error highlighting as you type**.

MOEDL (Metaheuristic Optimization Experiments Description Languages) is a domain-specific language for the description of metaheuristic optimization experiments (such as techniques comparison experiments, or parameter tuning experiments). Its goal is reduce the time and expertise required for describing those experiments. MOEDL documents are divided into three main sections: problems, techniques and configuration. The former includes details about the problem such as its type and problem instances to be solved. The second includes information about the metaheuristic techniques used to solve the problem, the termination criterion and random number generator used. The later includes information about the configuration of the experimental execution. A detailed description of the syntax of MOEDL and with several examples are provided in [2].

```
Labpack root
  Experiment Description
  input
      GeneralInputDataFile 1
      ...
  configurations
      config1
          input
              config1-InputDataFile 1
              ...
          executions
              exec1
                  exec1-OutputDataFile 1
                  ...
      artifacts
          src
          bin
          doc
  ...
```

Fig.3. SEA folder structure layout

Layout of experimental information (according to SEA). Experimental reproducibility requires not only providing a comprehensive and detailed description, but also providing all the input and output data of the experiment, and any experimental artefact used for its conduction, such as survey forms, data gathering spreadsheets, etc. The role of those elements in the experiment should have an impact on their location on the lab-pack, in order to ease its use. Thus, a generic default layout for the elements of lab-packs named SEA (Scientific Experiments directory lAyout) is described in detail in [2, appendix E]. EXEMPLAR supports the creation of SEA-compliant lab-packs, providing advices on where the

uploaded files should be located depending on the specific extension and role for such workspaces. Figure **3** shows the workspace creation form, highlighting the SEA layout option.

2.3 Automated analysis of experimental information

The killer feature of EXEMPLAR is its support for the automated analysis of experimental information. This feature enables the detection of: i) common validity threats in the experiments, and ii) inconsistencies between SEDL experimental descriptions and the actual contents of the lab-pack.

Automated detection of common validity threats. Currently, EXEMPLAR supports three operations that can detect common validity threats identified in the literature [3, 4, 5]:

Multiple Comparison: This operations checks if a single comparison statistical test is being used to perform multiple comparisons, leading to a statistical analysis validity threat [3].

Small sampling: This operation checks if the sample size of the experiment is sufficient for a safe application of the statistical tests specified for the analysis in its SEDL description, leading to a statistical analysis validity threat [3]. Currently, this operation checks a minimum sample size of 30 observations in null hypothesis statistical tests are applied.

Inconsistent Variable Measurement: This operation checks if the values of the experimental variables contained in the inputs and output datasets provided, are consistent with the corresponding domains specified in the SEDL description. Depending on the type of data inconsistency (missing value, value out of variable domain), and dataset role (input or output), this inconsistency can be symptomatic of a specific validity threat [2]. Authors plan to implement the whole catalogue of analysis operations described in [2] as future work.

Automated detection of missing experimental information in lab-packs. EXEMPLAR supports the checking of the consistencies between inputs, and outputs specified in SEDL experimental descriptions and the actual contents of the corresponding lab-pack. In this sense the repository can remind users to upload forgotten files, or find errors in the input/output files specification of SEDL descriptions.

3   From MOEDL to SEDL with MDE

The interpretation and analysis of MOEDL documents in EXEMPLAR are performed on the basis of its corresponding SEDL document. To that purpose, a set of transformation rules from MOEDL to SEDL has been created, i.e. any MOEDL

document can be automatically transformed to a SEDL document. This approach to the design of MOEDL has important advantages.

First, it enables the creation of more succinct experimental descriptions, since the elements that are common to any metaheuristic optimization experiment are skyped, and incorporated to the corresponding SEDL documents during the transformation process.

Second, this approach enables grouping several experimental design decisions into alternative choices in MOEDL reducing the risk of making mistakes when expressing the experimental design in SEDL. For instance, the transformation ensures that the statistical analyses specified in the transformed SEDL document are appropriate for the experimental design generated, based on the set of metaheuristics and problem instances to be compared.

Third, it allows the automated analysis of MOEDL experimental descriptions through the transformation using the analysis operations defined for SEDL documents.

4 Conclusions and future work

In this paper the main features of EXEMPLAR were presented. As future work authors plan to: improve the platform with template projects for labpacks, add auto-complete capabilities to the editors, and support the automated replication of in-silico experiments.

Acknowledgments

References

1. IEEE Computer Society: Software Engineering Body of Knowledge (SWEBOK). Angela Burgess, EUA (2004), http://www.swebok.org/
2. Parejo, J.A.: MOSES: a Metaheuristic Optimization Software Ecosystem. Ph.D.thesis, Univ. of Sevilla (2013), http://www.isa.us.es/publications
3. Shadish, W.R., Cook, T.D., Campbell, D.T.: Experimental and quasi-experimental designs for generalized causal inference. Houghton Mifflin, 2 ed. (2001)
4. Wohlin, C. and Runeson, P. and Höst, M. and Ohlsson, M. C. and Regnell, B and Wesslén, A.: Experimentation in Software Engineering. Springer (2012)
5. Juristo, N. and Moreno, A.M.: Basics of Software Engineering Experimentation. Springer (2010).