Reusing Legacy DSLs with Melange
Thomas Degueurle, Benoit Combemale, Arnaud Blouin, Olivier Barais

To cite this version:
Thomas Degueurle, Benoit Combemale, Arnaud Blouin, Olivier Barais. Reusing Legacy DSLs with Melange. 15th Workshop on Domain-Specific Modeling, Oct 2015, Pittsburgh, United States. hal-01197039v2

HAL Id: hal-01197039
https://inria.hal.science/hal-01197039v2
Submitted on 1 Oct 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Reusing Legacy DSLs with Melange

Thomas Degueule
Benoit Combemale
INRIA, France
firstname.lastname@inria.fr

Arnaud Blouin
INSA Rennes, France
arnaud.blouin@irisa.fr

Olivier Barais
University of Rennes 1, France
olivier.barais@irisa.fr

Abstract
The proliferation of independently-developed and constantly-evolving domain-specific languages (DSLs) in many domains raises new challenges for the software language engineering community. Instead of starting the definition of new DSLs from scratch, language designers would benefit from the reuse of previously defined DSLs. While the support for engineering isolated DSLs is getting more and more mature, there is still little support in language workbenches for importing, assembling, and customizing legacy languages to form new ones. Melange is a new language workbench where new DSLs are built by assembling pieces of syntax and semantics. These pieces can be imported and subsequently extended, restricted, or customized to fit specific requirements. The demonstration will introduce the audience to the main features of Melange through the definition of an executable DSL for the design and execution of Internet of Things systems. Specifically, we will show how such a language can be obtained from the assembly of other popular languages while maintaining the compatibility with their tools and transformations.

Categories and Subject Descriptors D3.2 [Language Classifications]: Specialized application languages

Keywords Domain-specific languages, language workbench, language reuse, model typing, melange

1. Introduction
Domain-specific languages (DSLs) are increasingly used to handle specific concerns in the development of complex software systems [1]. However, developing a DSL is still a costly and time-consuming task that requires advanced skills in language design: language designers have to specify the abstract syntax of their languages, their concrete syntax, semantics, and tooling (e.g. editors, checkers, code generators, etc.). Language workbenches assist language designers by providing the right tools and methods to tame the complexity of language design and reduce the development costs [2]. While current workbenches (e.g. MetaEdit+, Spoofax, MPS, Xtext – to cite just a few) propose a diffuse way to reuse language modules, there is currently little support for assembling languages with customization facilities. Yet, it is likely that the creation of new DSLs could benefit from the efforts spent on the development of other ones, especially when their domains overlap. A mere example is the family of statecharts languages which, despite their specificities, share many similarities in their syntax and semantics [3]. The expected outcomes are twofold: one would like to import the definition of legacy language artifacts to engineer a new one while ensuring the compatibility with the tools and transformations defined on its ancestors. Of course, imported artifacts may not fit exactly the designer’s expectations. It follows that support for language extension, restriction, and customization is required to tune them finely.

This paper is organized as follows. In Section 2, we introduce Melange, a new language workbench attempting to address each of these challenges. In Section 3, we present an outline of the proposed demonstration.

2. The Melange Workbench
Melange [4] is an open-source language workbench built on top of the Eclipse Modeling Framework (EMF) and tightly integrated with its ecosystem1. Thanks to the success of EMF in both academia and industry, this enables Melange’s users to import and manipulate a wide spectrum of existing DSLs. In Melange, the abstract syntax of DSLs is defined in the form of a metamodel with the Ecore formalism. Their operational semantics is specified using the Xtend programming language2. More precisely, Melange supports the definition of aspects which allows to define the operational semantics of the concepts contained in a metamodel in a non-intrusive manner, based on static introduction [5]. In Melange, pieces of syntax and semantics can be imported and assembled to form new languages. The resulting languages follow the same design principles: they consist of a metamodel and a set of aspects that can be directly bundled and deployed as is, or reused in other assemblies. In order to fit unforeseen requirements or new environments, Melange also provides customization operators: languages can be merged together, inherited, or sliced. Each of these operators takes both syntax and semantics into account. The merge operator serves as a language unification mechanism and is inspired by the UML Package-
3. Demonstration Outline

The demonstration will highlight the main features of Melange, illustrated through the creation of an executable DSL for the design and execution of Internet of Things (IoT) systems. The resulting IoT language is inspired from both general-purpose executable modeling languages such as fUML [10] and modeling languages dedicated to IoT such as ThingML [11]. We will show how the assembly operators of Melange foster the reuse of pre-existing languages. Specifically, the IoT language will be designed as an assembly of publicly-available DSLs: (i) an IDL language for specifying the structural interface of sensors (ii) Lua for expressing their behavior and (iii) an activity diagram to express concrete scenarios involving different sensors. Taken independently, each of these languages has been defined by different groups of people for specific purposes, unrelated to IoT systems. Combining them in a consistent way, however, leads to a new DSL particularly suited to a new context, i.e. the IoT domain. Because most of their syntax and semantics can be reused as is, this drastically reduce the development costs compared to a top-down approach. Of course, the syntax and semantics of two independent languages may not fit together perfectly when composed. Therefore, the demonstration will also focus on the customization of these reused languages. Finally, the demonstration will illustrate how the resulting DSL remains compatible with the tools and transformations (e.g. checkers, editors) previously defined on the imported languages.

References

[10] OMG. Semantics of a foundational subset for executable UML models (Fuml 1.0), 2011.
Figure 2: Assembling several variants of a finite-state machine language in Melange. The outline presents the abstract syntax of each language, the methods and runtime data inserted using aspects (e.g. a `fire()` method on transitions and the current state), and the subtyping relations linking their types. Language designers can thus build new languages on the one side and observe the results on the other side at the same time.