Conceptual Model of the Globalization for Domain-Specific Languages

Tony Clark, Mark van den Brand, Benoit Combemale, Bernhard Rumpe

Conceptual Model of the Globalization for Domain-Specific Languages

Tony Clark[1], Mark van den Brand[2],
Benoit Combemale[3], and Bernhard Rumpe[4]

[1] Middlesex University, UK
[2] TU Eindhoven, NL
[3] University of Rennes and Inria, FR
[4] Software Engineering, RWTH Aachen,

Abstract. Domain Specific Languages (DSL) have received some prominence recently. Designing a DSL and all their tools is still cumbersome and lots of work. Engineering of DSLs is still at infancy, not even the terms have been coined and agreed on. In particular *globalization* and all its consequences need to be precisely defined and discussed. This chapter provides a definition of the relevant terms and relates them, such that a conceptual model emerges. The authors think that this clarification of terms and the meaning will foster the field of efficient DSL definition and evolution in the future.

Keywords: Globalized DSLs, Conceptual model

1 Towards a Conceptual Model Of Globalization

Software Engineering, unlike other engineering disciplines, such as *Civil*, *Chemical* or *Material*, deals with constructing precise descriptions of highly complex systems, where each new application contains structure and behaviour that is essentially unique. In essence, each new application is a novel theory of structure and execution, and requires a way of expressing this meta-information [2]. Traditionally *General Purpose Languages* (GPLs) have been used to encode the theories in executable, but implicit forms (e.g., libraries). However recent advances in language engineering technologies have made it possible to develop *Domain Specific Languages* (DSLs) each of which is more suited to encoding theories relating to specific application domains [4].

Modern applications tend to be large, heterogeneous and distributed, involving the use of many different languages including mixtures of GPLs and DSLs. Given that an application consists of many different sub-systems written in different languages, there is a requirement to ensure that the languages and therefore the sub-systems work together effectively and must share the same concepts (theories). Sub-systems written in DSLs are attractive because the languages can provide better support for the specific application domains, however they tend to be less mature than their GPL counterparts and therefore there is an interesting research challenge: how to achieve language globalization [1] whereby GPLs and DSLs can co-exist and work together in order to achieve a high quality assured system.

DSLs introduce *meta-architectures* into the process of systems development. Naur [2] argues that systems development is the process of encoding theories about a specific system into GPLs. By the same argument, DSLs involve the process of encoding theories of a complete domain. Encoding a system theory into a GPL involves finding a way of mapping the theorems into the (often computationally-centric) domain supported by the GPL, even if this is done via libraries; whereas encoding a system theory into a DSL requires less cognitive dissonance. At least at the conceptual level, integration respectively globalization is achieved by finding mappings between the different theories that make up a system such that the mappings are maintained when mapped to the implementation languages.

A system that requires no integration with respect to globalization effort must be implemented in a single perfect DSL. Increased use of separate DSLs within a system will require mappings between the distinct theories, but will require no *implementation* mappings to be applied to the point-wise correspondences. This is possible if the DSLs are implemented using the same programming language or framework. A hybrid DSL/GPL system will need to deal with several such implementation mappings where the domain-specific nature of the theories has been lost through an implementation encoding; finally a GPL-only system must face a situation where all mappings are computationally encoded.

There has been very little exploration of the foundations and concepts that underpin methods and technologies needed to address the challenges of globalization such as those outlined above. The aim of this chapter is to perform a domain analysis for globalization such as defined in Definition 7 and to propose a conceptual model that can be used to organise and classify these challenges. In attempting to produce such a model we will encounter issues for which there are no current or no generally accepted solutions; these will be listed as research challenges at the end of the chapter.

2 Basic terms

Definition 1 (Model). *A model has three characteristics: There is an original that it models. The model is an abstraction with respect to the original. The model has a purpose with respect to the original. (Definition to Stachowiak, 1973) [3]*

Models are used in almost every science and engineering discipline for quite some time and for a variety of purposes. Some are prescriptive, where the model is developed before the system and used to describe and / or predict the systems properties. In natural and social sciences models are used to describe the systems under study (from subatomic particles to galaxies, from molecules to cells to animal behavior to societal behavior) and thus to understand (part) of these systems. It is important to precisely define the purpose of the model, in order to understand its appropriateness of the model as abstraction from the original.

The intended purposes are often clarified by the questions that a model should be able to answer.

Modelling is a rather old mechanism, computer science, however has made it possible that models are also shared between humans and computers, which led to the necessity to make modeling more explicit and more precise. Modelling languages necessarily emerged.

Definition 2 (Language). *A language is a means for communication between humans, machines, and humans and machines. A language describes the set of possible sentences that may be communicated between the stakeholders.*

Languages allow us to describe various things, among others expressing data (structure), computations (behavior), interaction, requirements, physical structure, networking structure, etc. As a consequence languages are amenable to both mechanical and cognitive processing. Usually sentences are handled as individually storable, versionable and manipulable artifacts. So it is legal to identify the sentence with the artifact that contains the sentence.

While the definition does not explicitly speak about language semantics or potential forms of use, a language normally also intends a semantics for its sentences as well as a pragmatics to clarify the forms of use.

Definition 3 (Domain Specific Language (DSL)). *A DSL is a language that is specifically dedicated to a domain of interest.*

DSLs are therefore typically restricted, both in the domain, where they are used and in their expressiveness. This on the other hand gives us the advantage to more easily design the language including syntax and semantics of all the language elements. A DSL should be seen in contrast to a general-purpose language (GPL) that is broadly applicable without any feature for a particular domain. In particular programming languages are typically GPL, but the Unified Modeling Language also is a general-purpose and thus domain agnostic language. If a DSL is used for modeling purposes, then we speak of a domain specific modeling language or DSML.

While a language set of sentences is usually infinite, we need a finite, comprehensive form to define a language. We distinguish the terms language and language definition, to make it precise what we are speaking about. There are many definitions for the same language and in the globalization context we manipulate language definitions, while defining new languages. For example the Java languages is a set of sentences (called programs) and can be described by a variety of mechanisms, including different forms of grammars.

Definition 4 (Language Definition). *A language is defined by the following concepts:*

- *Concrete syntax: e.g. in textual, tabular, or graphical form describing the set of sentences of the language.*
- *Abstract syntax: describing essential concepts and structure of the sentences without semantically irrelevant concrete sugar.*

– *Static semantics (or context conditions): Is a boolean predicate based on the concrete respectively abstract syntax. Sentences that fulfill the static semantics are called well-formed. They obey the context (scope, type system, etc.)*
– *Meaning (or dynamic semantics) of the sentences e.g. as operational, denotational, or axiomatic semantics.*

3 DSL Integration

A language that is used as the basis of mechanical processing consists of a collection of components including syntax, well-formedness checking, and semantics. Following Naur, we abstract from the implementation of a language to its definition, in which case the language consists of a collection of integrated theories. Each theory consists of theorems that relate to one particular aspect of the language, for example concrete syntax (its grammar), type checking, security, execution, memory usage, *etc.* In addition, a collection of mappings between the theories ensures that they work collectively to answer any question that is of interest relating to the language, for example linking information contained in a syntax-theorem to elements used in a type-theorem and a corresponding operational theorem.

The computational models used by GPLs have been under development for many years and are rather mature, though not really standardized. A number of meta-languages for expressing language-based theories have grown up around these models, for example λ-calculi, Hoare Logic, natural deduction systems, variants of states transition systems, petri-nets and pi-calculi. Deduction within these systems is well understood and general purpose, and it is possible to map from one to another.

Our conceptualisation should not be limited to those aspects of systems that are known to be supported by practical tools (parsers, type-checkers, compilers, and run-time systems). We wish to think of *any* aspect of a system being expressed using a theory that co-exists with all other aspectual theories so that an entire system including its history and its future is captured by a family of co-dependent theories. Therefore, theories relating to design co-exist with theories relating to security or privacy, and theories relating to distributed development can be related to theories about usability or hardware failure rates.

If a system is defined using these meta-languages then there is a good correspondence between GPLs and the resulting theories. However, following Naur, a system consists of a collection of domain-specific theories. For example a financial system may contain a theory about Sarbanes-Oxley or a pan-european education system may contain a theory about the Bologna Process. The domain-specific theories are not necessarily computational, but must be encoded within computational theories in order to achieve a practical system.

Encoding domain-specific theories within GPL computational theories, via libraries, leads to a problem that DSLs address. As an extreme example, consider a simple information system LibSys consisting of theories relating to library borrowing. The LibSys theories are not computational in the sense that they describe a step-by-step process, they simply define the arrangements of information that must hold and the services that are offered. In contrast, consider a machine that processes binary information in terms of a stack, a heap and a simple instruction set. There is a theory M that describes computations that are performed given specific starting states for the machine. Our challenge is to encode the LibSys theory into the M theory.

To understand and describe a language it is useful to regard it as a collection of inter-related theories which themselves are denoted in a collection of meta-languages, each of which is itself a language. This regress is usually grounded by using meta-languages that are well-understood and that do not require further elaboration. Introducing DSLs however means that there is an extra level of language definition that requires meta-languages to be defined on a per-application basis. This leads us to meta-meta-languages being the basis for definition, but also provides scope for the basic meta-meta-language to be fixed for all DSLs that are used in an application. Whilst this is not always possible, it is attractive because it facilitates the relationships between theories that are required for globalization.

4 Language Components and Interfaces

The integration of languages works best, when we use modularization techniques similar to those available for programming. We therefore propose the use of language components.

A language component captures all information about the language and exposes aspects to language users. Globalization is achieved by mapping between aspects in terms of the concrete interface data. The idea of language definition liberates us from having to say how the language is implemented and also how the interfaces are achieved.

Definition 5 (Language Component). *A language component (aka. language module or language unit) is a reusable encapsulation of a, possibly incomplete, language. A language component includes a language definition and might include explicit provided and required language interfaces.*

A language component may be incomplete in three ways: First it may be parameterized, such that other language components can be plugged in. Second the language component may itself be dedicated for composition and thus not be of (much) use as standalone language. That also means that each language itself is a language component, that however is free of parameters and complete in that sense that it can be used purposeful.

Third, a language may be incomplete in its components. While the main ingredient of a language, namely the abstract syntax must always be given, a language component may omit the concrete syntax or a definition of the semantic domain and mapping. For an engineering point of view, it may also omit editor, compiler, generator, and other operational realizations useful for a language. We assume that language adaptations take place to add missing constituents of a language.

Globalization of components enforce interfaces, where components are glued together.

Definition 6 (Language Interface). *A language interface is a relevant abstraction for a specific purpose of a provided or required part of a language component. An interface can be defined manually in a separate artifact or inferred automatically from the language component definition.*

This is a very general description for language interfaces that will have different characteristics dependent on languages that are interfaced and the purpose of the composition. It may be a syntactic interface, connecting syntax, may be an interface describing imported and exported types, variables and other kinds of names, or maybe a purely semantic interface allowing to connect the semantics of language components. Technical interfaces also may connect editors, analyzers, synthesizers or code generators for the respective language components.

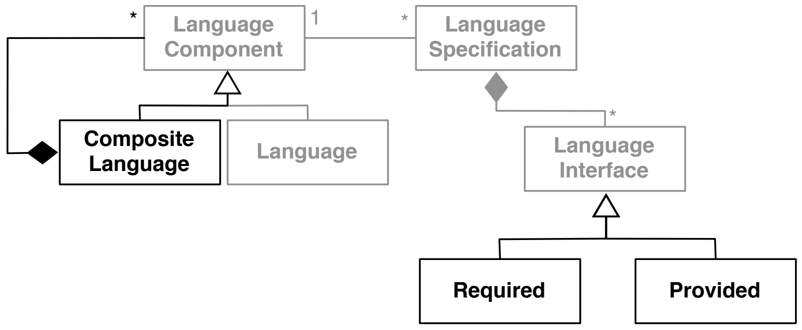
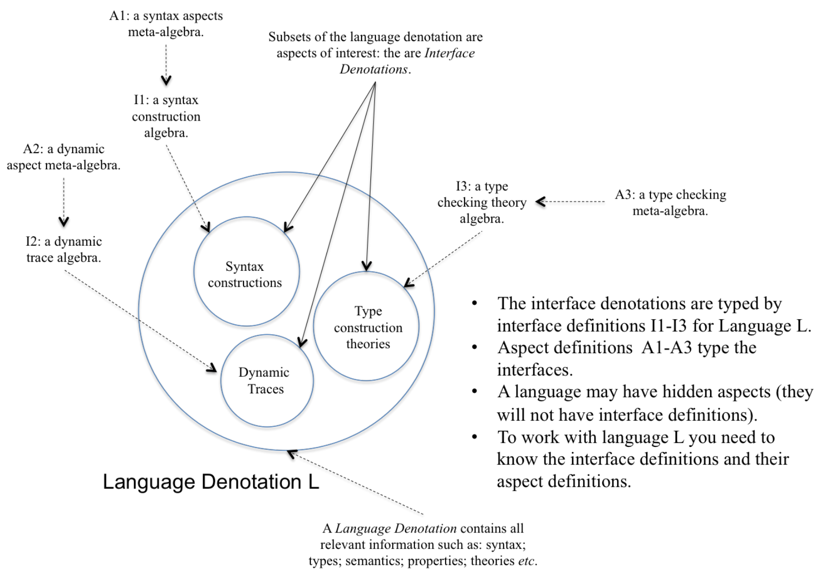


Fig. 1. Language Component (open question!)

Figure 2 gives a simple view of globalization in terms of *language definitions*. A language definition captures the information about a language in a technology independent manner, *i.e.*, without recourse to implementation technologies such as parsers, compilers, run-time systems *etc.*

The idea is that a language definition is the essential characterising information about a language. All aspects of a language are captured in principle, however some will be of more use than others with respect to globalization. For example, if globalization is to occur exclusively at run-time then the syntactic definition of a language (the set of program phrases) is of no interest; whereas, if an editor is to be used to integrate two or more languages then syntax may be the only aspect of interest.


A1: a syntax aspects meta-algebra.
Subsets of the language denotation are aspects of interest: the are *Interface Denotations*.
I1: a syntax construction algebra.
A2: a dynamic aspect meta-algebra.
I3: a type checking theory algebra.
A3: a type checking meta-algebra.
I2: a dynamic trace algebra.
Syntax constructions
Type construction theories
Dynamic Traces
Language Denotation L
A *Language Denotation* contains all relevant information such as: syntax; types; semantics; properties; theories *etc*.

- The interface denotations are typed by interface definitions I1-I3 for Language L.
- Aspect definitions A1-A3 type the interfaces.
- A language may have hidden aspects (they will not have interface definitions).
- To work with language L you need to know the interface definitions and their aspect definitions.

Fig. 2. Language Components and Interfaces

5 Globalization

Definition 7 (Globalization). *Globalization deals with the purposeful construction, adaptation, coordination and integration of explicitly defined languages, to be amenable to mechanical and cognitive processing, with the goal of improving quality and reducing the cost of system development.*

Globalization is achieved (partially or wholly) in terms of syntactic and/or semantic integration from the perspective of globalization stakeholders. Globalization addresses all aspects of the life-cycles of languages and the systems developed with them and therefore affects their development and coordination of multiple concerns, methods, documentation, tools, variations and maintenance. Language globalization affects several levels, such as type or meta-type, and may be achieved statically or dynamically, or a mixture of both.

It is useful to be aware of the different stakeholders and especially their individual and therefore often conflicting goals and backgrounds.

Definition 8 (Globalization Stakeholder). *Any person who is affected by the definition or use of a language or its components is a globalization stakeholder.*

Globalization stakeholders include the globalization strategist who is responsible for the globalization strategy for an organisation; the language engineers that develop or prepare languages to be globalized; language integrators that

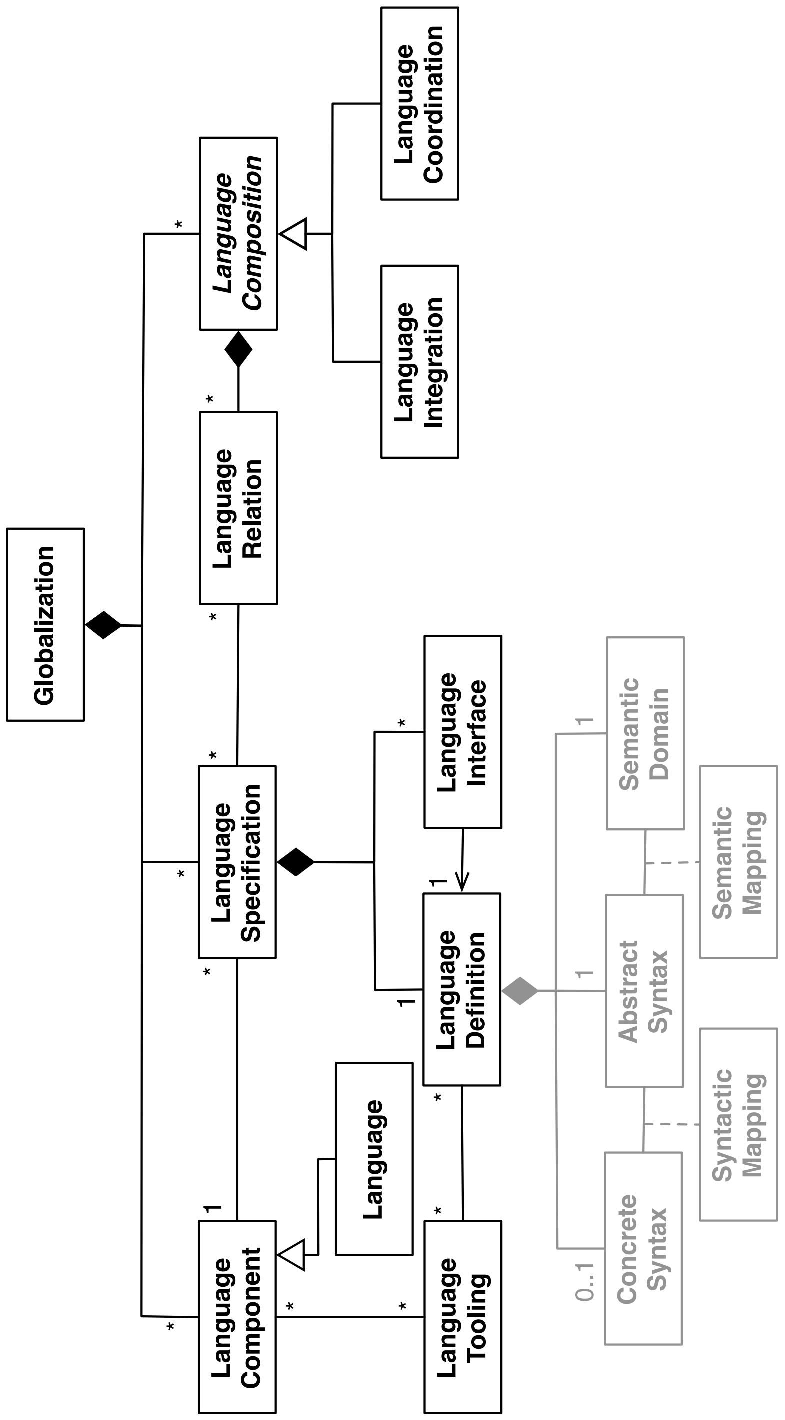


Fig. 3. Conceptual Model for the Globalization of DSLs

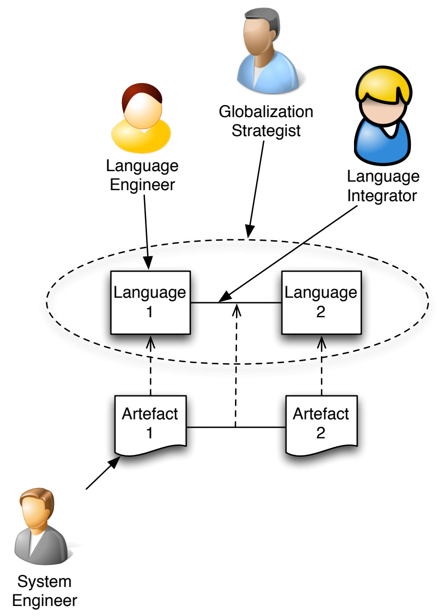


Fig. 4. Globalization Stakeholders

ensure that two or more languages are globalized; software and system engineers that use a collection of globalized languages for manipulating artifacts.

Globalization will involve at least two but usually more language definitions. Integration will occur in terms of the information contained in different aspects. The information used for integration is defined by the language interface since they will be used in mappings between languages. For example, if two different languages have type systems then they must expose their respective type definitions to a mapping that associates types from one language to corresponding types in the other.

Globalization is *specified* in terms of language definitions and the specification of relations between them. Once globalization is specified, it must be implemented. Clever implementation techniques must be found that achieve efficient mappings between the required aspects.

6 Language Relations

We examine the various existing forms of relationships between languages:

Definition 9 (Language Relation). *A language relation relates the sentences of multiple languages.*

In simple cases, only two languages respectively their sentences are related. However, it may be that in a relation sets of sentences are related to each other on both sides. This captures e.g. composition and slicing as relations.

If a relation needs to be effectively executed, an algorithmic mapping is necessary to realize the relation:

Definition 10 (Language Mapping). *A language mapping is a language relation that has an algorithmic, effectively executable realization that maps sentences of the source languages to sentences of the target languages.*

As many languages do have infinitely many sentences, a relation has to be expressed in a finite form for example using the language definitions. So instead of relating entire sentences of the languages, usually concepts of the syntax structure or the semantic structure are related. The relation can range from purely syntax based to a relation between the semantic domains. Several interesting cases are:

– If some of the syntax constructs and the corresponding semantics of the languages are identical the relation for these constructs is the identity. It is sufficient that the abstract syntax is identical, the concrete syntax may differ.
– If the abstract syntax constructs differ but the semantic domains are identical, then there is no need for a semantic integration anymore. For an even tighter coupling a syntax based relation can be added, that is consistent with the semantics relation.
– If both differ syntactical and semantical relations can be provided.
– The case that the abstract syntax is the same, but the corresponding semantics differs should be avoided, because this leads to unsolvable problems. However, it may be that the abstract syntax is the same, but the semantics encoding differs even though the intended meaning of both languages is the same. In this case a relation of between the semantics (respectively their encodings) should be provided.

The complexity of the semantic relation depends on the "distance" between the semantic domains of both languages. While we don't feel able to fully define the term "distance" here, we might agree that the larger the "distance" between two languages is, the deeper the "encoding" of one language in the other needs to be. E.g. state machines can be "deeply" encoded to a relational database schema, by encoding their entire syntactic structure using a state and transition table, while a relation between class diagrams and relational data base schemata can be relatively "shallow". If may even be not feasible to define a semantic relation. The syntactic relation may be influenced by the fact that information has to be removed or created.

The type of mappings used in globalization between language definitions is defined in terms of the types of the constituent interfaces. For example, if two type-systems are defined using different meta-languages then the mapping is a relation between the meta-languages. The types of the interfaces are called *aspects* of the language. As noted above, some aspects may be of more importance to globalization than others, and some aspects may be limited to the construction of the definitions, *i.e.*, hidden to any external language user.

7 Composition

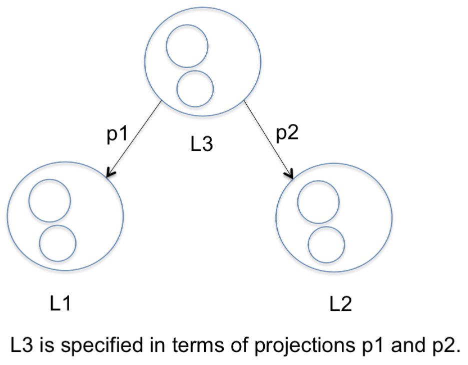


Fig. 5. Language Composition

Definition 11 (Language Composition). *This is an abstract concept that achieves globalization in terms of multiple languages working together to achieve a common goal.*

Composition may be achieved using a number of strategies including language integration and language coordination. The composition of two or more languages may require additional information in the form of a correspondence between the syntax and/or semantics of the constituent languages. A language can be decomposed to produce two or more languages in which case the decomposition is to be viewed as the inverse of the corresponding composition.

Our claim is that globalization requirements can be conceptualized in terms of language definitions, their interface definitions and associated aspect definitions. A globalization requirement for a set of language definitions S involves the specification of a new language definition L such that a collection of constraints holds between L and S. The language L is the required globalization language.

Consider two languages L1 and L2 and a requirement to globalize with respect to both syntax and operational semantics. This might be expressed as the construction of a new language definition where the globalization requirement is expressed in terms of the syntax and operational semantics interface definitions (that are required to be present by reference to the associated aspect definitions). Such a requirement might involve the definition of two language transformations p1 and p2 that are defined in terms of the interface definitions and capture the syntax and operational semantics of the language L3 as shown in Figure 5.

Language components contain the definitions of interfaces that correspond to theories about aspects of the languages that are required for globalization. As such, language components are themselves structured elements that can be subject to transformation and combination with other language components. Therefore, we envisage a calculus of language module construction, combination and transformation operators that characterise globalization.

8 Language Coordination

The coordination of several languages is a special, loose form of composition.

Definition 12 (Language Coordination). *Language Coordination is a form of composition where individual sentences of the coordinated languages are collaborating to achieve a common goal.*

As a consequence of this definition is that, while the languages are coordinated for a specific purpose, the artifacts (containing the sentences) of the coordinated languages remain as individual artifacts. This allows to independently reuse them, and include artifacts of each of the languages in the coordination.

Coordination implies that the controlled languages remain independent. For example, dynamic coordination might be achieved by registering an observer with two independent run-time systems that propagates changes from one run-time to another. Language coordination is to be contrasted with language integration where two or more languages are merged to produce a new language.

Coordination can be achieved via sharing concepts with the same semantics. The corresponding models do not exchange information explicitly, but reasoning about artifacts related to shared semantic concepts becomes easier. Coordination can also be achieved via sharing of concepts with different semantics. The corresponding models have to exchange information explicitly. Tools that manipulate the models should provide facilities to exchange of information. The information can data or control based.

As an example, consider a globalization requirement for two DSLs. The first DSL L1 manages a data-base and provides a collection of event-based rules. Events occur when data changes. Rule-actions can cause further updates to the data. The second DSL L2 defines simple form-based input screens. Buttons can occur on forms. The language component for L1 provides an event aspect and a data aspect. A hidden aspect of L1 is a sequence of event-driven data-base traces. The language component for L2 provides a button-press aspect and a form-content aspect. A hidden aspect of L2 is a button-press driven sequence of form-traces. The mapping that specifies the globalization of L1 and L2 associates the events from L1 with the button presses of L2 and the data states of L1 with the form-content of L2.

The implementation of the globalization specification may require some form of common data representation to be defined to that the form information is available to the data-base when the event is raised. In addition, a communication mechanism must be implemented that ensures an event is raised in L1 when a button is pressed in L2. There are many such implementation architectures that would be consistent with the globalization specification.

9 Language Integration

Another major form of language composition is the integration of two languages:

Definition 13 (Language Integration). *Language Integration is the production of a new language from a set of individual languages.*

The resulting language has a its own set of sentences, but each sentence has "sub-sentences" which come from the individual sub-languages. For modeling languages we also call those constituencies "model components" in correspondence to the language components.

An integrated language is not required to exhibit all language concepts of its sublanguages. For example, state machines and Java might be integrated to produce a new state machine language that uses Java statements as actions and Java boolean expressions as guards.

Language integration has been studied since it became clear that the definition of new languages is complex and error-prone. Language integration is the type of form of the reuse of individual components and heavily relies on a crisp and well-defined notion of language interface, because this is the place where languages are syntactically integrated and where static semantics as well as dynamic semantics has to conform.

10 Towards the Conceptualisation of the Globalization of DSLs

This article defines a number of terms in an abstract way for dealing with the globalization of DSLs and relates them in various ways. However, in practical use there are pretty many open questions, to answer.

1. What is a language interface?
2. What is a language component?
3. What is language composition?
4. How to use a language interface?
5. How to facilitate language integrations?
6. How to facilitate language coordination?
7. Are there other forms of language composition?
8. What is an appropriate formulation for language relation?
9. How does a language relation relate to a language concept relation?
10. Can we identify general mechanisms for language composition or is composition highly specific to syntax, semantics and purpose of languages?

The key to globalization is the ability for all stakeholders to understand how the interaction between models can be facilitated via the relationships between the constituent languages. It would appear that a fruitful way to achieve this is to apply a component based approach to language composition. Such an approach implies a clear definition of language component and language interface, however this is an open research question at this stage. Therefore, an important area for future research should be to conceptualise language components and to propose concrete mechanisms for component and interface definitions. For example, should interfaces be models? How can existing languages be wrapped to produce components? How can components be linked together via interfaces? What should a provided and required interface for a language be? If interfaces are models are there such things as meta-interfaces? Can interfaces provide access to all levels of language (instance, model, meta-model)?

System developers within a globalization context should be, as far as possible, unaware of the integration machinery when developing their models. This is a significant research challenge for tool developers. In addition to tooling, a new methodology for MDE may be required in order to guide globalization stakeholders.

In addition to the primary concepts defined in this chapter, the following issues are potentially relevant to successfully achieving globalization: language libraries; language viewpoints; sub-languages; language transformation and adaptation; language construction; language and system quality. We do not have definitions for these terms in the context of globalization and therefore they should be considered as areas for further work.

References

1. Benoit Combemale, Julien Deantoni, Benoit Baudry, Robert France, Jean-Marc Jézéquel, and Jeff Gray. Globalizing Modeling Languages. *IEEE Computer*, pages 68–71, June 2014.
2. P. Naur. Programming as theory building. *Microprocessing and Microprogramming*, 15(5):253–261, May 1985.
3. Herbert Stachowiak. *Allgemeine Modelltheorie*. Springer-Verlag, Wien, New York, 1973.
4. A. Van Deursen, P. Klint, and J. Visser. Domain-specific languages: An annotated bibliography. *ACM Sigplan Notices*, 35(6):36, 2000. Disponible en http://homepages.cwi.nl/ paulk/publications/Sigplan00.ps.