Approaches to IDE Context Assistance for Complex Domain Specific Languages

C.Grieve

Introduction

A common usage of domain specific language (DSL) is the Extended Mark-up Language (XML) where the domain is specified by some form of schema.

This brief paper attempts to summarise and extrapolate on one particular area of experience gained over a four year period of designing schema configurable editing tools for complex language domains.

Approach

As an aid to authoring a complex DSL such as that required by a microprocessor or similar highly configurable device, an authoring IDE will typically prompt the user with the available language commands for a particular domain context. This prompt is usually in the form of a drop down list or context menu of available options. There are several levels of implementation sophistication for this:

Implementation 1: The Big Dumb List and It's Variants.

Presents the user with a context independent list of all the nodes available for the domain.
The advantages of this approach are:

- It is easy to implement
- Is of great assistance to the expert user

The disadvantages are:

- Allows the user to create invalid content.

A slightly more sophisticated version of the above presents the user only with those choices that might descended from his current context. Again this has the advantage of a simple implementation but the disadvantage that it allows the user to create invalid content.

Implementation 2: The Context Sensitive List

It is possible to analyse a DSL schema to present the user with only the options that would result in valid content. The advantage of this approach is that:

- The user is only presented with choices that would result in valid content.

The disadvantages are:

- It is more difficult to implement.
- In a deeply nested complex schema, a high level user may not be familiar or care about the intermediate elements of the specification but be interested only in inserting particular key commands, unavailable from many contexts.

Implementation 3. Back to the Big Dumb List

The user is presented with all of the nodes available as descendants of that context, but where a node requiring other siblings or descendant nodes to be present is inserted; the IDE automatically inserts the node(s) required to produce valid content. Advantages are:

- The user is again only presented with options that would result in valid content.
- The ability to create heavily nested nodes is catered for, freeing the user of knowledge of the complexities of the domain

The disadvantages are:

- It is more difficult to implement
- Low level options, irrelevant to the high level user are still presented. The high level user is not completely abstracted from the complexities of the domain.

Conclusions

The most productive implementation of the simple drop down helper list would appear to be implementation 3. However, the lack of abstraction of this method for the high level user is a cause of concern. Possible solutions to this problem are:

- Creation of a separate abstract DSL schema that could be translated to the low-level DSL.
- Some technique of annotating the low-level DSL schema so that the options associated with the higher level tasks could be separated and presented to the high level user.

As a general conclusion, where high and low level tasks need to be performed within the same domain space some method should be used to segregate the context assistance presented to these differing types of user.