# Semantic Document Structuring and Sustainable Development in MMiSS

Bernd Krieg-Brückner

Bremen Institute of Safe and Secure Systems, Universität Bremen,
Postfach 330440, D-28334 Bremen
`bkb@Informatik.Uni-Bremen.DE`

**Reusability and Extensibility.** One major problem with the preparation of teaching materials of any kind is that the adaptations necessary for each teacher and each year make reuse of older material almost impossible. It is often necessary to completely restructure a course to integrate new developments and results into hand-outs and overheads. Thus a major goal of the MMiSS system is to guarantee users the highest degree of flexibility, extensibility and reusability of the materials stored within it. The author of content should be able to assemble teaching packages from a structured system of individual units and atoms, by using the structural and semantic relations explicit in the representation, such as pre-conditions, cross-references, related units and alternatives. Thus content can be prepared by different people with different goals and requirements, but together and as part of the same repository, possibly in different variants and views. It should be easy for teachers to combine different materials, even from different authors, into a whole, and for students to use alternative material.

**Extensible Knowledge-Base.** The speed at which knowledge develops is a special problem in Computing Science. It is imperative to be able to continuously extend and modify the stored knowledge. This leads to consistency problems; for example, it must be clearly specified whether a cross-reference (hyperlink) refers to the newest version or to some specific older version; these could be lost or outdated because of modifications to the referenced content. This is especially important in the context of Formal Methods: a referenced definition must fit into the application context which may not necessarily be compatible. For example, the name of the term defined by the other author may be different, or, worse, the other author may use the same name for an entity that has a subtly different semantics. The system to be constructed shall solve this by keeping track of semantically different entities, independently of their apparent name in a specific context, and by storing additional meta-data in the knowledge-base. Semantics and functionality of knowledge are to be clearly separated from representation.

**Semantic Relations in the Development Graph.** As an example of a structure, a section of a document may contain mathematical definitions, theorems and proofs (with connecting texts); a similar situation arises with overhead transparencies for lectures. Texts contain embedded *formal* components (theorems, formulas, proof-scripts) which can of themselves be processed by corresponding systems. A textual nesting (the *"is contained in"* relation) yields at first a tree structure. This is extended by *semantic relations*, defined explictly by the user or implicitly by the system. For formal components this structure is evident; a formula representing a theorem to be proved *lives in* a theory; a proof-script that proves this theorem within a proof-system is subordinate to this theorem, or in general to a relation containing a proof obligation (*proves*). In the course of development, alternatives arise, for example an alternative proof in the example; this, like earlier versions, must be preserved so that it is possible to return to it. Thus there is in general a *Development Graph* containing related units and the developments leading to a particular configuration.

**Methodology for Sustainable Development.** Semantic approaches from software engineering and, in particular, Formal Methods, can contribute to solve the problem of how to continuously develop and maintain multimedia educational content. The development methodology of *(stepwise) refinement* is already known; this could for example be applied to working out the materials with more precision or in more detail. The important point is that this activity of refinement is preserved in the development structure. Another concept borrowed from Formal Methods is the so-called *conservative extension*, which preserves the original content so that a reference to the extended version remains valid for the original meaning. This concept has a well-defined verifiable semantics, which naturally cannot be guaranteed for textual, or other multimedia, content. Thus consistency must be preserved through discipline among the developers, rather than formal proofs. In any case, a *real* change forces all dependent content to be reworked (this can be automatically recognised and communicated to the authors), while a conservative extension does not.

**Variants,Version and Configuration Management.** An important dimension for Development Graphs is that of versions and their administration. Usually only the *current version* is of interest and all earlier ones are not shown; however, an option should make alternatives to a version visible. It should be possible to select between *variants*, such as the language used (for example "British English" or "German"), the formalism (for example "CASL" or "CSP"), or the level of detail (for example "Lecture Notes", i.e. overheads augmented by comments and explanations, suitable for study after presentation in class), and so on. The notion of a *consistent configuration* is important here; all objects related to a particular selected object should belong to the same version, or at least to a semantically compatible one. The document actually displayed (a subgraph) should in a well-defined sense be *complete*; thus when for example the formalism"CASL" is chosen as a variant, examples should generally be available in the formalism"CASL", and similarly when a particular level of detail is chosen. It is then possible to freeze a configuration as a publicly-accessible *edition.* All these functions, especially verifying consistency and completeness, should be supported by the system.