"compound" document definition

XHTML, SVG and SMIL are modularized languages which, in principle, could share some common modules. Hence one could imagine to create a new modularized language built from the list of modules already used inside XHTML, SVG and SMIL languages. In the following sections below, we don't adopt this approach and we assume that XHTML, SVG and SMIL keep their current descriptions and thus, we only describe our view about the problems raised by integrated XHTML, SVG and SMIL sections inside a unique document.

Should there be a set of predefined compound document profiles?

There is a need for defining classes of applications in order to recommend the best combination of languages for each, without defining any specific profile for each language itself (which is done in the recommendation of the later language).

For each class of application, the recommendation would describe the following :

- What are the main features of each class in term of evolution in time and space, rendering and functional features ?
- which language must be the host language (main document) and why ?
- which document type may be embedded or linked to the main document and how ?

So far, we would like to introduce 3 predefined classes of applications, which can be extended of course during the workshop :

- time-centric applications :
  - the main key parameter of the document is time.
  - Examples : slideshows, animations like cartoons or greeting cards, video synchronized with text
  - The recommended host language would be SMIL (or one of its profiles)
    - Why : because SMIL is already recommended for MMS and particularly well suited for synchronizing medias in time
  - The possible embedded languages would be XHTML, SVG, and/or TimedText as rendering languages of some regions of the SMIL presentation/animation

- browsing-centric applications :
  - the main feature of the document is text/image layout which doesn't evolve with time except in some areas of the document which size is fixed by the main document.
  - Examples : information service including mainly text and images about weather, news, …
  - The recommended host language would be XHTML (or one of its profiles) with only text and images allowed
    - Why : because most existing applications of this class are based on this language today and XHTML remains the best known language by authors
  - The possible embedded language would be SVG only as the Rich Media rendering language of some regions or images. Since SVG 1.2 integrates most of SMIL modules, embedding pure SMIL documents would have no added value. Idem for TimedText (to be analyzed deeper).

- Vector-graphic-centric applications :
  - the main feature of the document is vector-graphics layout which doesn't evolve with time except for some areas of the document which size is fixed by the main document. For this class of application there is a need for at least one of the following functionnalities :
    - being able to Zoom-in/out the main content or adapt the orientation and size of the content according to the screen of the device capabilities
    - being able to draw shapes like circles, triangles, …which can be filled/stroked
    - being able to support complex path definitions
    -
  - Examples : location-based applications, …
  - The recommended host language would be SVG (or one of its profiles)
    - Why : because SVG is particularly well suited for describing vector-graphics scenes

- The possible embedded language would be SVG (if scalable text fonts are needed) or XHTML (if pre-defined system fonts are sufficient). Since SVG 1.2 integrates most of SMIL modules, embedding pure SMIL documents would have no added value. Idem for TimedText (to be analyzed deeper).

This list of classes is not exhaustive but aims at covering 90 % of the current use case needs. Other classes could be introduced in order to meet other specific needs :

- simple games (quiz)
- chat applications
- 3D applications

As we can see, the idea IS NOT :

- to define and recommend subsets of each language (profiles) adapted to the application, which should be rather decided in device-oriented specific organizations like OMA
- nor to propose a unique set of tags which solves some redundant features of the different languages (both XHTML and SVG 1.2 describe different mechanism for defining forms for instance). Because W3C must take into account the existing implementations (players and authoring tools) and the fact that most content authors are already used to edit languages like XHTML and it would take too long for them to learn a new language rather than using an existing one.

The first idea is rather, for each class, to have an homogeneous set of recommendations on which language to use and rules concerning styling, event handling and scripting in the compound document.

The second idea is that, if for some reason one functionality of a class can be rendered by 2 languages, only one language must be recommended for this class of application to render this functionnality. For instance, if it is possible to show that most TimedText features can be done with latest version of SVG 1.2, there is no need to allow TimedText content to be embedded in a browsing-centric application since SVG is already allowed and able to do that.

What happens with event processing and style cascading across the boundaries of mixed content?

The rules we would like to support are that all styling (CSS) and scripting is defined and managed in the host document (i.e. the main document). Neither scripting, nor styling is allowed in the inserted documents which means, for instance, in the case of time-centric application that no <script> or <style> tag would be allowed in the embedded SVG or XHTML documents.

Concerning event handling, it is easy to define a “bubble” mechanism : an event (like ‘onBegin’, ‘onClick’, …) is first transmitted to the lowest level of the compound document. If it is not processed there, it is then forwarded to the parent document, and so on … Finally, the event is handled in the main document by ECMAScript or some other scripting mechanisms.

What features are needed from authoring tools designed to generate mixed content?

The ideal authoring tool would of course manage the notion of class of applications and profiles for each hosted language. Which means that the author is first asked (via a Wizard) to select the class of application he wants to design and the profile of each component languages for this class.

The authoring tool would manage the compound document in a global way, which means that it manages all the aspect of the edition of the document (timeline, layout, layers, …) and doesn’t call a specific authoring tool for each embedded document of the main document.
In other words, the main document and its embedded contents are seen as a whole and the user is able to access several views of it :

- Wysiwyg view (timeline and/or layout depending on the class of application the user selected)
- Source code view

- DOM view : evolution with the time is shown when the user selects a position in the timeline of the main document (if it's a time-centric application) or a component document (if it uses SMIL/SVG/TimedText or any other timed language).

The authoring tools manages style and scripting and puts all the generated content in the <head> or <header> tag of the main document. It is not possible to put CSS style or ECMA script function in a specific element, especially if this element belongs to a component document.

The main advantages of this approach are :
- there is no need to define a common interface between authoring tools
- the rules concerning scripting, styling and event handling are simpler to implement
- once the user has chosen a class of application, he is not allowed to put many heterogeneous formats in the main document

The main drawbacks of this approach are :
- need to re-implement already existing authoring tools which were dedicated to a specific language
- component documents can not be re-used easily in other main documents

What MIME type should a compound document use?

If the notion of class of application is introduced, it is easy to define one MIME type for each class as follows :

application/compound-doc-time-centric
application/compound-doc-browsing-centric
application/compound-doc-vectorgraphics-centric

How can application semantics from different markup languages be mixed in an interoperable way (e.g. using XBL)?

This is a very risky subject. The main risk if this matter is treated in a generic way would be that no one would be able to implement it nor to create any content with it.

The best approach according to us is to define a new class or to extend an existing class of applications if there is a specific need for embedding a new language in the main host document (like TimedText or Xforms for instance).

Is there a need for a generic extension architecture? What is needed to allow extensions, such as plug-ins, to handle content that is not supported directly by the browser/host environment?

The idea we support is to have an integrated Rich Media player and authoring tool, which are able to play and create one or several class(es) of applications. In this condition, there would be no need to define an extension mechanism.