Definition and Generation of Self-contained Digital Forms based on the OASP4JS Reference Architecture

The work "*Self-contained Digital Forms based on Web Browser Technologies*" proposed a format for digital forms which is based on common web technologies like HTML and JavaScript. In the following, this format will be referred to as *DForms*. The goal of DForms was to provide a platform-independent format for digital forms which can easily be integrated into existing processes and allows for a high level of usability.
In contrast to web forms, which usually need to be deployed on a web server and store input data remotely, DForms work like any local application, storing data locally. The advantage of DForms over usual desktop applications is that they do not require a separate installation since the web browser, which as of today is present on most modern devices, serves as the runtime environment.

However, one major requirement for digital forms could not be satisfied yet: the ease of creation. Simplicity was an important factor motivating DForms as a format for digital forms. But in the current state only developers with deeper knowledge in the format, the used technologies and frameworks are able to create their own DForm. DForms are missing a reasonable level of generalization regarding reusable form components and tools supporting their creation.

This is the starting point for the definition and generation of DForms. The goal of the following work is to enhance DForms in such way that common forms can be created without requiring advanced programming skills. This goal should be achieved by following a declarative approach for defining DForms that is mostly independent of the underlying programming language and facilitates comprehensive tool support.
Digital forms are heavily characterized by their contained data, which suggests describing them with respect to these data and the desired constraints on them. Such a description can be thought of as an abstract schema definition of a digital form and it builds the basis for generating an executable implementation. This implementation should be compliant to the *OASP4JS* architecture. Since this includes a Client-server model, DForms may also be extended to be more flexible with respect to switching between a full Client model and a mixed Client-server model. Since the form definition is mostly independent of this aspect, the form generation may take the two different use cases into account. This would for example allow to start with a local DForm version and later on integrate it in a Client-server system, which includes server-side form execution and validation.
The generation process requires mapping the schema elements to form elements and arranging the form elements in a graphical layout. Further application logic that has no direct relation to the data may also be added. In order to simplify the generation process, there should be a framework that provides common form elements and an interface to the data.
However, the described generation process should not hinder form creators from realizing more complex digital forms. For this reason, it should be possible to create custom extensions for the generation framework, e.g. realizing specific form elements, and to implement advanced application logic that may not be completely covered by the declarative form description.
Both the definition and the generation process should be accompanied by a graphical user interface that abstracts to a certain degree from the underlying programming languages and opens up the creation process to form creators with no or only little experience in the used technologies.

Following the conceptual considerations, a prototype will be implemented that covers the complete creation process including the definition and generation of DForms. This prototype will be used to evaluate the results of this work by analyzing the feasiblity of several typical use cases for digital forms.