Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 4 (2011) 668–677

Procedia Computer Science

International Conference on Computational Science, ICCS 2011

IODA - an Interactive Open Document Architecture

J. Siciarek, B. Wiszniewski*

Faculty of Electronics, Telecommunications and Informatics, Gdansk University of Technology, Narutowicza 11/12, 80-233 Gdansk, Poland

Abstract

Objective of the proposed architecture is to enable representing an electronic document as a multi-layered structure of executable digital objects, which is extensible and without a need to support any particular formats or user interfaces. IODA layers are intended to reflect document content organization levels rather then system abstraction or functional levels, as in software architecture models.

Keywords: document engineering, multi-layered architecture, interactive service composition

1. Motivation and background

Since the beginning of human civilization documents have been widely accepted as natural means for representing information and providing interfaces to human driven processes. Today, when it is strightforward to store and process them as any other digital data, an important issue is not to offer electronic document solutions forcing human users to abandon their habits developed with traditional paper-form documents. Although notations for representing a document in a computer memory, or tools for manipulating its content, may change, simple actions like annotating or marking document pages directly on their image is the most natural and expected functionality of any electronic document system. Surprisingly this is not an easy task given the myriad of formats used to represent a document in a digital form needed for its storage: from binary bitmaps of document scans, through description of document pages in postscript, or editable word processor content, up to logical document structure descriptions with typesetting or markup notations. Many of these formats require different tools to view a document content in a form resembling its hard copy image. Web browsers attempt to bridge that "format" gap by utilizing the mechanism of plugins, but marking a visual content of thus rendered document is not easy due to the problem of robust anchoring, needed for marking and annotating a document at the client side – regardless of a document format and functionality of the server side [1, 2]. A limited solution might be providing functionality at the document server side to enable marking and annotating its content by remote users. It should however be independent of a document format to provide forward compatibility with standards that may appear in the future, as well as escape from the trap of developing a heavy machinery for "programming" papers that may quickly become obsolete before reaching maturity.

A perspective of associating markings of a document content with functions leads to a tempting concept of *executable documents*. This concept will certainly extend a traditional understanding of an electronic document by

*Corresponding author
Email addresses: siciarek@wp.pl (J. Siciarek), bowisz@eti.pg.gda.pl (B. Wiszniewski)

Selection and/or peer-review under responsibility of Prof. Mitsuhisa Sato and Prof. Satoshi Matsuoka
doi:10.1016/j.procs.2011.04.070

leveraging its usability as an interface unit to services doing anything anywhere on the Web. Interactive Open Document Architecture model proposed in this paper provides a solution to the problem of making a document content executable despite of its particular representation format. The proposed solution is light-weight, i.e., does not require documents to be in any particular format, relies as much as possible on the existing technologies and tools and takes advantage of interactive interpretation of a document content by human users rather than automatic parsing with a hard wired code.

2. Multi-layered document architecture

A key point about IODA is that it does not introduce any specific document format nor require implementing document functionality in any particular programming language. It just provides a sort of a document *spine* that binds tools and services that already exist in a Web document ecosystem. A document spine is implemented as an XML file that simply combines three layers shown in Figure 1.

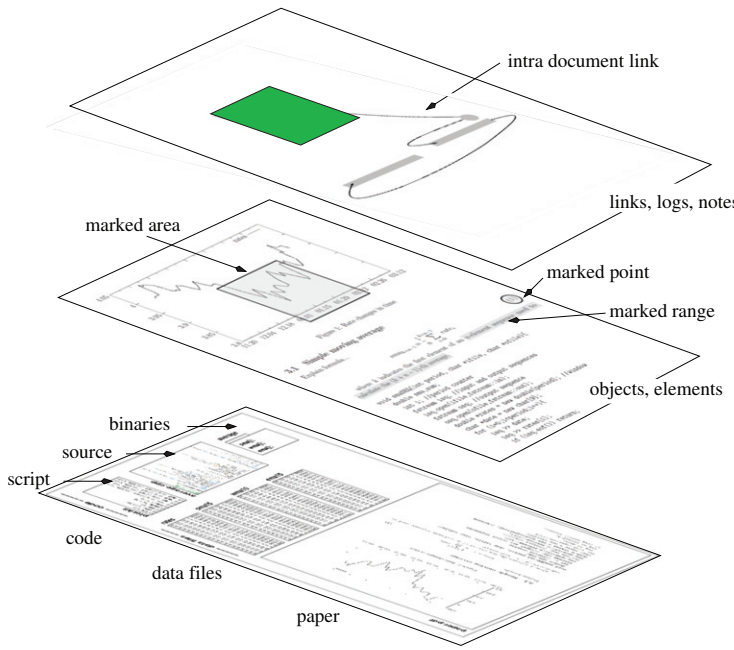

Figure 1: IODA layers

Layers of IODA reflect broad interpretation of a "digital object" from a viewpoint of digital libraries [3]. A *data layer* collects bits and bytes of text and binary files, which are just recorded "facts". Patterns that underlie those data and enable their interpretation constitute an *information layer*. Finally "contexts" of the interpretation patterns constitute a *knowledge layer*.

2.1. Data layer

This is the base layer and contains a *main document* in a form of a file ready to display or print. Document content may be binary, e.g. an image in TIFF or JPEG, or textual, e.g. PDF or RTF. Along with the main document file the data layer may contain or refer to text or binary files with data, scripts, source code, executable binaries, and so on.

Binary files and scripts implement *services* that can make document content executable. IODA distinguishes three kids of such services called respectively: *embedded*, *local* and *external*. Embedded services are executed at the host document server for data stored in the data layer. Local services are performed as browser *plugins* or default *local applications* at the client side – upon downloading data from the data layer. Further modifications of downloaded data do not affect the original content of a document data layer, as they are performed at the client's workstation and