Ecosystem infrastructure for smart and personalised inclusion and PROSPERITY for ALL stakeholders

D202.1-2 Report on repository standards, common interfaces and APIs (2nd iteration)

Project Acronym	Prosperity4All
Grant Agreement number	FP7-610510
Deliverable number	D202.1
Work package number	WP201
Accessibility, Standards, Developers, Components, Programming Interfaces	Building Blocks
Authors	Till Riedel, Gregg Vanderheiden, Martin Deinhofer, Lukas Smirek, Simon Bates, Matthias Berning, Chris Veigl
Status	Final
Dissemination Level	Public
Delivery Date	31.10.2017
Number of Pages	29

Keyword List

Accessibility, Standards, Developers, Components, Programming Interfaces

Version History

Revision	Date	Author	Organisation	Description
11891	04/12/2014	Till Riedel	KIT	Initial Outline
12054	15/12/2014	Gregg Vanderheiden	RTFI	Edited Strategy
12154	18/12/2014	Martin Deinhofer	FHTW	Links added
12321	21/01/2014	Lukas Smirek	HDM	Input on various Standards
12522	29/01/2014	Simon Bates	IDRC	Input on Meteor
12577	02/02/2014	Matthias Berning	KIT	Input on Mobile Accessibility
12583	02/02/2014	Chris Veigl	FHTW	Input on Bio Signals
12584	02/02/2014	Till Riedel	KIT	Complete Draft Version
12864	02/02/2014	Till Riedel	KIT	Review Marco Aimar
13084	02/02/2014	Gottfried Zimmermann	HDM	Review
20940	25/09/2017	Gregg Vanderheiden	RTFI	Added link to DeveloperSpace
23513	25/10/2017	Till Riedel	KIT	Added content for 2nd iteration
23519	30/10/2017	Till Riedel	KIT	Documented more use of standards)
23520	31/10/2017	Till Riedel	KIT	Spell Checks

Table of Contents

1 Executive Summary

The DeveloperSpace is now live at **https://ds.GPII.net**. It contains not only the work done on Components as part of the Prosperity4All project but also hundreds of components from the open source community.

This documents defines the purpose of adopting and referring to standards within the GPII Developer Space Component Listing. We focus on existing standards, that are used by components or will guide the continuing convergence of components particularly within the http://ds.gpii.net. The latest update of this report update the actual use of standards within the open source community developing accessibility technologies with special focus on the parts created within this project.

NOTE: It is very important to note that the use of any standards is in no means mandatory for the inclusion of content to the http://ds.gpii.net. This list is purely informative only gives a first guidance on what standards to consider and when to use them.

2 Standards and common APIs as prosperity strategy http://ds.gpii.net Building Blocks

2.1 Introduction and Methodology

The Global Public Inclusive Infrastructure (GPII) uses standards wherever possible and appropriate. Because the GPII is open by definition, use of standards facilitates its function both by making it easier for others to contribute to the effort, and by making it easier for other technologies to work with the GPII. The GPII both uses established public standards and creates its own internal standards for these purposes. The team working on the GPII also contributes to multiple international standards efforts that relate to its work (Report P4A/D503.4 contains information).

2.2 Purpose and structure of this document

The purpose of this document is to collect and document the different standards being used in the http://ds.gpii.net that host potentially very heterogeneous technologies.

The first part (section 1) of the document explains the role of standardization and particular types of standards for the http://ds.gpii.net.

The following parts (section 2-4) list different areas of standardization that are currently important within the http://ds.gpii.net. This section has a reference character, but also should serve as basis for discussion on the scope of use of the mentioned standards. In the latest update it will contain information on the factual relevance of the components.

Both parts are living documents that will be maintained throughout the project and are viewable and editable via http://wiki.gpii.net/w/Developer_Space/Standards on the GPII Wiki.

The view on concrete standards changed as expected drastically during the project time. As already predicted in the first release of this document in 2015 developments within the GPII and different interactions (such by the Prosperity4All Implementations) but also external developments shaped the landscape. This second edited revision this document reflect those developments.

2.3 Purpose of the http://ds.gpii.net

With the http://ds.gpii.net we collected components, frameworks, and tools that can make it easier to create new accessibility solutions both as assistive technologies, and as features to mainstream products.

The Prosperity4All http://ds.gpii.net also now provides as planned an amount of community resources and documentation that will help foster the adoption of accessible and flexible components.

The Developer Space is “The one-stop place to find resources, components and people to conceive, develop, test and market novel accessible solution” (see http://ds.gpii.net/)

2.3.1 Current state of the http://ds.gpii.net

The http://ds.gpii.net is now live and open to wide audience. During the past 2 years Prosperity4All boot-strapped by creating a semi-structured listing of components (that started at http://wiki.gpii.net/w/Developer_Space/Components) and resources (http://wiki.gpii.net/w/Developer_Space/Resources) on the common wiki with upcoming search functionality. In addition, discussions between developers and potential implementers have started via the mailing list (http://lists.gpii.net/cgi-bin/mailman/listinfo/dspace). Rather than being a fixed entity, the Developer Space was an evolving structure within the GPII that actively brought all kinds of component developers and product implementers together, which are interested in assistive technology and inclusive design.

This work lead to the current implementation that is converging towards the evolving but also stable, reliable platform, that we currently have. Importantly the development of technologies for aggregating information from existing repositories like github.com, particularly the component listing has grown considerably from just tens of items to hundreds of different components. This current brought list of components, that was released initially in September 2017 build the basis for the analysis of this report.

2.3.2 Use of Standards and common APIs within the http://ds.gpii.net

Not all of the components that will be in the http://ds.gpii.net will have standard interfaces. Some of them are just code snippets or technologies developed by others that we have brought into the http://ds.gpii.net.

Standards in this context can be existing standards or well-structured and well-documented common APIs (that can also be considered standards, see below). This convergence through standards and APIs is an important step towards the objective of cutting down the cost of improving the inclusiveness of IT and adapting new technologies for AT.

As it turns out there are also only few standardized or stable APIs in the domain. Most components that we are seeing, that are actively developed, change interfaces quite rapidly. Changes are mostly only documented within the code or in usage example in README files. As the by far largest number of resource particularly components targeting web developments, are loosely coupled.

2.4 Possible reasons to not to standardize existing components

Standards typically comprise formal documentation for common (technical) interfaces, requirements and practices. Often, however, particularly in the internet domain, existing products or conventions between multiple parties become so accepted that they can be considered “de facto standards” even without the necessary formality. The reasons for

using standards are many and well-documented (e.g. see http://www.iso.org/iso/home/standards/benefitsofstandards/ , http://archive.webstandards.org/edu_faq.html or http://wayback.archive.org/web/20081207115434/http://www.sutor.com/newsite/essays/e-OsVsOss.php). Particularly accessibility is an important aspect of many standards found in the web. Standards are nowadays essential parts of many ecosystems where many stakeholders are involved. Nonetheless we want to note potential reasons, why we may not use "real" standards for things within the http://ds.gpii.net or in special cases even not adapt to what is considered "de facto standard".

- **"No existing standard is a good match for what is being done"**

A developer may not want to make the effort to adapt his work to something, that reduces effectiveness or functionality by forcing a fit to an existing standard meant for something different or more limited.

- **"Standardized niche products are still niche products"**

Niche products that need a lot of other adaption anyway (not out of the box usable) won't profit from standards. Other strategies, such as opening and modularizing the code might be better strategies of technical adoption.

- **"Standardization does not increase the reach and value"**

The applicable standards are not used widely enough in any target market to make the standardization effort worthwhile. On the other hand, standards might force you to make decisions that do not allow you to support or focus on a niche group. The actual reach and value of a component might actually not increase if it is standardized, but will become "yet-another" piece of software that attaches to a bigger effort.

- **"Standardization and markets are not consolidated enough"**

APIs are trending and fast moving, so are new standardization areas driven by multiple stakeholders and standardization organizations. If committing to one standard binds a lot of resources, it might be better to wait to see which community or vendor(s) take up the product, and identify which standards they would like to use, before investing in standardizing component interfaces and focus on functionality.

- **"Providing standardized interfaces on component level is not a value in itself"**

Ripping components out of an existing implementation in order to make them reusable is not a value in itself if there is no identified general use (or standard) for the component. Here, also other strategies like documenting may be of more benefit to those that would like to incorporate a component in their product.

Before considering or promoting standards these risks should be analysed, documented and addressed. For this it is critical to reflect how much the use of a standard can contribute towards a goal, like wider adoption and sustainable development.

As the GPII supports the use and distribution of Open Source software it naturally also strongly encourages the use of Open Standards. Standards should most certainly not be used if they limit, rather than encourage, the distribution of components. Openness is an essential aspect of the http://ds.gpii.net and should be supported by appropriate standards.

In the scope of different standardization organizations (such as the ITU-T or the IETF) or different national governments (such as the Danish "Definitions of Open Standards", 2004) or even laws (the Spanish "Ley 11/2007" of Public Electronic Access of the Citizens to the Public Services, 2007) the definition has been different. Today no general definition is established within the European Union while it at least gives a definition in the scope of

eGovernment Interoperability (European Interoperability Framework for pan-European eGovernment Services, Version 1.0 (2004) ISBN 92-894-8389-X):

- ''The standard is adopted and will be maintained by a not-for-profit organisation, and its ongoing development occurs on the basis of an open decision-making procedure available to all interested parties (consensus or majority decision etc.). ''
- *The standard has been published and the standard specification document is available either freely or at a nominal charge. It must be permissible to all to copy, distribute and use it for no fee or at a nominal fee.*
- *The intellectual property - i.e. patents possibly present - of (parts of) the standard is made irrevocably available on a royalty-free basis.*
- *There are no constraints on the re-use of the standard.*

Particularly the last two bullet points differ from many other definitions that focus mostly on transparency. Similar definitions have been published by others, such as the Open Source Initiative (http://opensource.org/osr). Further reading may include Ken Krechmers “The Meaning of Open Standards” (http://www.csrstds.com/openstds.html).

2.5 Types of Standards and common APIs to consider

2.5.1 Guidelines and Requirement Catalogs

Many standards in accessibility are guidelines to support developers ensuring standardized definitions of accessibility. Governments and legislature often specify broad functional accessibility requirement that particularly apply for public services to set minimum standards that are particularly relevant for end products. Detailed requirement sets might be standardized, as in ETSI EN 301 549 "Accessibility requirements suitable for public procurement of ICT products and services in Europe". As with other guidelines, suppliers or producers of content or technical solutions may declare conformity with a complete standard or parts of it. In contrast to the above mentioned standards, those do not deal with technical interoperability. Particularly mainstream implementers will look for certain components to ensure conformance with functional accessibility requirements. An important question in the scope of the http://ds.gpii.net is if functional requirements can be addressed or guaranteed on component level at all.

2.5.2 Technical Interoperability Standards

A particular subdomain is AT interoperability standardization that should support meeting accessibility guidelines within a certain IT domain. Those standards rather define interfaces than functional requirements. Particularly for systems that - unlike the Web - do not allow dynamic rendering of user interfaces, need specialized interfaces for accessing lower level functionality via accessibility APIs. As one example, ISO/IEC 13066-1:2011 is a standard for designing and evaluating interoperability between IT and AT.

It describes requirements for IT and AT products in three types of interoperability: hardware-to-hardware, hardware-to-software, and software-to-software. It also identifies a

variety of software-to-software APIs that are described further in other parts of ISO/IEC 13066. These APIs can be used as frameworks to support IT-AT interoperability. IAccessible2, an extension of the Microsoft Active Accessibility API designed by IBM and now an open standard, is one of the APIs described in ISO/IEC 13066. (http://www-03.ibm.com/able/open_computing/iso.html)

2.5.3 De facto Standards and APIs

As seen by the aforementioned examples, many interoperability standards were originally driven by platform vendors as part of their operating systems, such as the Microsoft Active Accessibility, and UI Automation Specification, adopted by Microsoft products, the Apple Accessibility Toolkit or the Assistive Technology Service Provider Interface, mostly used by open source operating systems and particularly free desktops. Different APIs within the concrete GUI toolkits implement them either implicitly or with the need of explicit developer support (adding extra information). The IAccessible2 Standard, curated by the Linux foundation and based on Microsoft's Active Accessibility, is an example that shows the importance of market penetration for the success of standards.

2.5.3.1 Operating Systems

Desktop environments expose many interfaces that are relevant in our everyday work and personal life. Since the advent of the personal computer usability of those systems is an important factor for a large part of the population. According to internet user statistics by www.netmarketshare.com Microsoft Windows (91.45%) dominates the market with Apple MacOS (7.21%) and different flavors of Linux (1.34%) splitting up the rest of the market. The mobile operating system market in contrast is, according to the same source, split equally between Android (45.86%), developed mostly by Google on the basis of a Linux Kernel and iOS (43.15%), leaving Windows Phone only (2.28%) behind Symbian (4.62%) and JAVA ME implementations (2.92%).

Supporting particularly one of the major operating systems and their APIs can be considered a de facto standard. On the other hand, standards that are not well established within the ecosystem of one of those operating systems will have difficulties. While aforementioned efforts within ISO/IEC 13066 to standardize and converge AT interoperability has progressed to some degree, the accessibility APIs for mobile devices are still driven mostly by the vendors. While for desktop operating systems AT interoperability between operating systems is far from perfect, despite standardization efforts, more and more accessibility features are included in mobile operating systems. This is particularly interesting since the graphical user interface components are far more consistent on mobile devices so that once accessibility interfaces are implemented they may work for a wider share of applications (implicitly).

2.5.3.2 Frameworks, Architectures and APIs

Particularly in web-based applications underlying platforms play a much smaller role compared to desktop applications (for both server and client). Presentation and

communication interface is widely standardized by Web Standards (see below). While standardized browsers and web views are available for most platforms, diverse web frameworks that are used for content creation and rich web applications have become de facto standards beyond formal standardization. Often frameworks today are compiled of a small core and so-called plugins, modules and widget that are often developed by a wide range of third parties. The framework core can be seen as standard for the whole ecosystem. Examples are jQuery or angular.js. This is different from classical cross-platform GUI-Toolkits like QT or GTK+, that have a different type of development community.

One interesting development are cross-platform layers for web-applications like Apache Cordova or so-called Polyfills (http://en.wikipedia.org/wiki/Polyfill), which try to support functionality before formal standard adoption. Both make particular use of HTML5 and JavaScript technology to provide a more uniform API landscape even before final standardization.

The role of the JavaScript as “the assembler language of the web” is further manifested in many server side APIs built in JavaScript via the node.js framework. Component frameworks developed within this community are often far less standardized than for example the dynamic component system defined by the OSGi Alliance for Java. Efficient execution of scripting languages together with platform as services providers, that run the code on web servers, has led to diverse ecosystems “in the Cloud”. Particularly the lifecycle management for many applications has been shaped by cloud computing on one hand and mobile apps on the other.

Today this often means much tighter interactions between components than foreseen by many of the standards that were driven by service oriented computing paradigms. Nonetheless service interfaces e.g. defined by the OASIS play an important role particularly in different business domains.

“The Hypermedia as the Engine of Application State” proposed by Roy Fielding is becoming more a reality than fixed, standardized interfaces. Today many interfaces reference to REST as a standard: GET, POST, PUT and DELETE in combination on resources identified by uniform resource locators (often in combination with JavaScript Object Notation or eXtensible Markup Language payloads) often serve as a common denominator in many applications today (not only in the Web). More and more formal standards define functionality on those minimal interactions. Particularly in new frontiers for IT systems like Smart Homes or the Internet of Things, REST Services together with other Web Technology are picking up. Looking at the fact that accessibility played an important role in the World Wide Web from the beginning this can be seen as an important opportunity to provide accessible interfaces also in a “Web of Things”.

Today also, a number of digital assistant platforms (in contrast to classical assistive technology) from other domains like Home Automations, Internet of Things and Multimedia become prominent de-facto standards with high practical relevance. This includes platforms like Google Home, Amazon Echo/Alexa, Apple HomeKit/Siri. Although their primary target market is not assistive technology targeting disability, their ecosystem often provide high value in terms of inclusion giving either access to many devices in an accessible manor or providing alternative input modalities such as speech. Although comparable to operating

systems (next section) they are basically often only comprise a set of open APIs that vendors need implement for interoperability that however than mostly need to interact with proprietary cloud based systems.

2.5.3.3 Use within the Developer Space

Components in the Developer Space largely follow de-facto standards and use the existing abilities to interoperate with other components in an opportunistic fashion. Mouse and keyboard emulation in both hardware and software still plays an important role. However with the rise of touch enabled interfaces or non-desktop interfaces, particularly web technology seems to be considered the most future proof way to enable alternative input and output modalities to applications. Thus we are seeing a move to web based interoperability as the main driver for accessibility. This is reflected by the fact that more that twice as many components are developed in languages like HTML, CSS and JavaScript that are rather associated with web development rather than in languages like C, C++, Python and Java. Even many components written in non-web languages are often back-ends for web enabled applications, that provide machine learning or media transformation pipelines for transforming content.

During the course of the Prosperity4All project many components were converted to support web based APIs. E.g. the AsTeRICS framework can now be controlled in a pure web based fashion. The WebACS uses a new REST based API. This API also makes it possible to build other web base applications on top of AsTeRICS, greatly opening the framework to other use scenarios. While the Integrated Runtime Environment demonstrates this integration with frameworks like MyUI, GPII Auto-Personalization and URC, the real value lies not in the even more extensive framework, but in the ability to recombine with other frameworks beyond the mentioned ones, that was enabled by that development. With the use of mostly HTTP/REST interfaces also in parts, the decline of monolithic frameworks has begun. E.g. with much overlap in functionality the Universal Control Hub that implements the full URC specification has already partially superseded by implementations that use existing technology like Eclipse Smart Home or the newly developed OpenAPE server over REST interfaces. The ability to cherry-pick components based on real need has been a major change during the course of the past years. Instead of providing interoperability into one direction, multiple direction can be taken, e.g. AsTeRICS can now communicate with a web browser, either directly or via the NEXUS middleware. Integration patterns and examples are provided rather than full architectures as a response to the need of implementers that are working in a brownfield environment.

In some cases even no servers are needed at all. Lately with the release of the Adaptive Web Components many core concepts of MyUI could be converted to latest web standards interfaced through JavaScript (also using the above mentioned polyfills). This makes them interoperable with many other components that are web based and listed in the Developer Space. Furthermore frontend developers do not have the need to adapt their server infrastructure, potentially leading to much faster rollout and adoption. The agile aproach that emerged during the Prosperity4All project complements particularly other necessarily much slower standardization work inside the Global Public Inclusive Infrastructure, that will in the end profit from deep interoperability in a number of application once e.g. an auto-

personalization infrastructure (as prototyped in the Cloud4AllProject and currently being rolled in pilots) is globally available to users.

Within the GPII there are still many ongoing and long running standardization activities. Following a “eating your own dog food” approach, interoperability with those standards is particularly promoted within GPII associated projects. However, the Developer Space has extended well beyond those standards and has become a representation of the current ecosystem reflecting needs of (particular commercial) implementers and other stakeholders in the accessibility domain.

3 Categorizing Functionality, Requirements and Preferences

An important factor for the Developer Space will be that implementers find useful tools and that developers can offer their tools in a fashion that they are found specifically by stakeholders in the accessibility domain. Therefore it is wise to look particularly at categorization schemes. While this might be not so important for mainstream software developers looking for concrete functionality, many non-classical developer types we want to address, like AT product developers, healthcare professionals or relatives have a domain specific view. Furthermore procurement might be a driving factor for accessibility improvements and the need for adoption of components, so that alignment with existing schemes might prove as an opportunity in some cases.

3.1.1 Assistive products for persons with disability -- Classification and terminology (ISO 9999:2011), Global Medical Device Nomenclature (GMDN) and Systematized Nomenclature of Medicine – Clinical Terms (SNOMED CT)

The Global Medical Device Nomenclature (GMDN) is a comprehensive system of internationally recognized coded descriptors in the format of preferred terms with definitions used to generically identify and characterize types of medical devices.

ISO 9999 is another standard particularly focused on assistive products. It offers both classification and terminology for specific AT product functionality and also classification means for adaptive technologies and products. The standard is maintained by the ISO and is continuously updated to meet technological developments and new needs. Every assistive technology is assigned a six digit code within a three stage product hierarchy.

The following table shows the top level classes.

Assistive product hierarchy according to ISO 9999

Assistive product hierarchy according to ISO 9999

top class	name	defined subclasses
Class 04	Assistive products for personal medical treatment	15 subclasses
Class 05	Assistive products for training skills	11 subclasses
Class 06	Orthoses and prostheses	12 subclasses
Class 09	Assistive products for personal care and protection	19 subclasses
Class 12	Assistive products for personal mobility	14 subclasses
Class 15	Assistive products for housekeeping	5 subclasses

Class 18	Furnishings and adaptations to homes and other premises	11 subclasses
Class 22	Assistive products for communication and information	13 subclasses
Class 24	Assistive products for handling objects and devices	13 subclasses
Class 27	Assistive products for environmental improvement, tools and machines	5 subclasses
Class 30	Assistive products for recreation	11 subclasses

The WHO has made efforts to map those classes to diseases and functioning profiles (http://whqlibdoc.who.int/hq/2010/WHO_HSS_EHT_DIM_10.2_eng.pdf) to identify Priority Medical Devices (PMD) based measuring the burdens of diseases world wide. Also, the International Health Terminology Standards Development Organization (IHTSDO) has set out to improve health of humankind by providing standardized clinical terminologies. The Systematized Nomenclature of Medicine – Clinical Terms (SNOMED-CT) even provides formal, logic-based definitions based on a multilingual thesaurus to be processed electronically. SNOMED-CT provides an overlapping hierarchical scheme and triple relations that enable description logic based reasoning. It also provides a subdivision of assistive products that however, does not directly map ISO 9999. More detail, particularly on ISO 9999, is given in the International Encyclopedia of Rehabilitation.

3.1.1.1 Use within the Developer Space

This categorization scheme was a candidate ontology for categorizing components in the Developer Space. With relation to components in contrast to end user solutions, this standard plays practical no role for current software developers outside of the pure health domain. Components are no products may be used in many classes. The GPII is further mostly focus to provide access to ICT as are most of open source components we could find. A categorization that would add most components would fall into Class 22 provides little value.

3.1.2 User Profile Management (ETSI EG 202 325) and User Profile Preferences and Information (ETSI ES 202 746)

A set of ETSI standards specifically aims at user profile providers, operators, service developers, service providers, device manufacturers and standards developers. The ETSI EG 202 325 standard describes how profile information should be managed. “Special needs” regarding accessibility are seen as part of such preferences. The ETSI ES 202 746 standard specifies information and preferences, which are choices made by the user, that will result in driving the behaviour of the system, and builds on the user profile concept described in ETSI EG 202 325. Profile solutions following the standard should be provided for the primary benefit of the end-user and give end-user the rights to manage the profile contents. It is interesting because it addresses auto-personalization but acknowledges particularly the right for informational self-determination.

3.1.3 Accessibility guidelines for information/communication technology equipment and services (ISO 9241-20)

ISO 9241, which addresses "Ergonomic requirements for office work with visual display terminals", contains in its part 20 a set of guidelines that can help to ensure that ICT equipment can be used by the widest range of people, regardless of their capabilities or disabilities, limitations or culture. It follows a very holistic understanding of inclusive IT that covers disabilities from birth as well as for elderly people, temporary or situational disabilities, and aims at guaranteeing ergonomic use regardless of physical, sensory and/or cognitive impairments.

3.1.4 Information technology — Individualized adaptability and accessibility in e-learning, education and training (ISO/IEC 24751)

This standard applies to personalization on demand enabled by ICT and networked systems distinguishing:

- Display: how resources are to be presented and structured;
- Control: how resources are to be controlled and operated; and,
- Content: what supplementary or alternative resources are to be supplied.

Similar to ISO 9241, it is not restricted to classic notions of disability and takes an integrative approach. It is intended for all users as every user can experience a mismatch of their individual needs and preferences and the content or services delivered. While the first part defines a framework, particularly the second part is interesting in the scope of categorization. This categorization focuses on the functional abilities and the assistive technology or other non-standard technology in use and distinguishes itself clearly from medical approaches and standards that focus on naming and describing human impairment. The standard provides attribute descriptions and usage recommendations for common adaptation preferences.

ISO 9241 is currently being revised with input from GPII. The common terms registry is a major contribution of the GPII to that effort that implements a software component for managing common terms. The required fields and procedures entry and maintenance of the Common Terms Registry are being defined by the new ISO 24751. The needs and preference set used for auto-personalization build upon those common terms to let the user specify what they want/need/prefer the environment to look or behave like. Consequently adopting common terms will help components of the Developer Space particularly interact with the infrastructure components provided by the GPII. Either adopting or mapping of component preferences to terms will enable auto-personalization based on the existing matchmaking features. Further information on the standardization can be particularly found on the AccessForAll Working Group wiki.

3.1.5 IMS Global Access for All Information Model Data Element Specification and Accessibility Web Schemas

The IMS Global Learning Consortium specifies a formal information model that adapts Access for All specification. IMS has licensed these properties to schema.org under the Creative Commons Attribute-ShareAlike License so that they can be used under schema.org's corresponding terms and conditions. The a11ymetadata website links applications of the specification and its relationships with previous metadata efforts.

The interesting part of the effort to align with schema.org, a general metadata tagging effort launched by major search engine operators, is that accessibility requirements can be used as search filters by custom web searches. Examples include particularly filtering online videos by captioning. Early adoptions demonstrate how to describe resources particularly, how meta-data can be included into learning resources in different scopes. An example that is particularly relevant for the media transformation components in the Developer Space is tagging of accessible media in terms of accessibility features, hazards and control functionality as visible by using the google richsnippets viewer on selected examples from the accessible online library bookshare.

Schema.org primarily uses the WHATWG MicroData specification to embed ontologies directly into web content. The vocabulary builds primarily on existing standards and best practices. In contrast to other efforts, it has been supporting multiple syntaxes, specifically including RDFa and JSON-LD (http://www.schema.org/docs/faq.html). This particularly allows embedding content from very different rich web applications and mash-ups that may mix different vocabularies. More information can be found via the W3C page on Accessibility WebSchemas.

3.1.6 Accessibility requirements suitable for public procurement of ICT products and services in Europe (ETSI EN 301 549)

ETSI EN 301 549 is a catalog of "functional accessibility requirements applicable to ICT products and services". Among other aspects, it also elaborated related test descriptions and evaluation methodology to a level of detail compliant with ISO/IEC 17007:2009. It aims primarily at public procurers to standardize the requirements for their call for tenders and at manufacturers to address standardized accessibility requirements within their design, build and quality control. The EU mandated to European Standardization to actually standardize requirement lists for certain ICT products on the basis of this standard. One of the outcomes of this mandate is a generator for standardized call for tender texts (http://mandate376.standards.eu).

A generated tender for cloud computing applications will contain text like: The offered solution shall meet Clauses 5.2, 5.3, 5.4, 5.5.1, 5.5.2, 5.6.1, 5.6.2, 5.7, 5.8, 5.9, 6.2.1.1, 6.2.1.2, 6.2.2.1, 6.2.2.2, 6.2.3, 6.2.4, 6.3, 6.4, 6.5.2, 6.5.3, 6.5.4, 6.6, 7.1.1, 7.1.2, 7.1.3, 7.2.1, 7.2.2, 7.2.3, 7.3, 9.2.1, 9.2.2, 9.2.3, 9.2.4, 9.2.5, 9.2.6, 9.2.7, 9.2.8, 9.2.9, 9.2.10, 9.2.11, 9.2.12, 9.2.13, 9.2.14, 9.2.15, 9.2.16, 9.2.17, 9.2.18, 9.2.19, 9.2.20, 9.2.21, 9.2.22, 9.2.23, 9.2.24, 9.2.25, 9.2.26, 9.2.27, 9.2.28, 9.2.29, 9.2.30, 9.2.31, 9.2.32, 9.2.33, 9.2.34, 9.2.35, 9.2.36, 9.2.37, 9.2.38, 9.3, 10.2, 10.2.1, 10.2.2, 10.2.3, 10.2.4, 10.2.5, 10.2.6, 10.2.7, 10.2.8, 10.2.9, 10.2.10, 10.2.11, 10.2.12, 10.2.13, 10.2.14, 10.2.15, 10.2.16, 10.2.17, 10.2.18, 10.2.19, 10.2.20, 10.2.21, 10.2.22, 10.2.23, 10.2.24, 10.2.25, 10.2.26, 10.2.27, 10.2.28, 10.2.29, 10.2.30, 10.2.31, 10.2.32, 10.2.33, 10.2.34, 10.2.35, 10.2.36, 10.2.37, 10.2.38, 10.2.39, 10.2.40, 11.2, 11.2.1.1, 11.2.1.2, 11.2.1.3, 11.2.1.4, 11.2.1.5, 11.2.1.6,

11.2.1.7, 11.2.1.8, 11.2.1.9, 11.2.1.10, 11.2.1.11, 11.2.1.12, 11.2.1.13, 11.2.1.14, 11.2.1.15, 11.2.1.16, 11.2.1.17, 11.2.1.18, 11.2.1.19, 11.2.1.22, 11.2.1.23, 11.2.1.24, 11.2.1.25, 11.2.1.26, 11.2.1.27, 11.2.1.28, 11.2.1.29, 11.2.1.30, 11.2.1.31, 11.2.1.32, 11.2.1.33, 11.2.1.34, 11.2.1.35, 11.2.1.36, 11.2.1.37, 11.2.1.38, 11.3.2.1, 11.3.2.2, 11.3.2.3, 11.3.2.5, 11.3.2.6, 11.3.2.7, 11.3.2.8, 11.3.2.9, 11.3.2.10, 11.3.2.11, 11.3.2.12, 11.3.2.13, 11.3.2.14, 11.3.2.15, 11.3.2.16, 11.3.2.17, 11.4.1, 11.4.2, 11.5, 11.6.1, 11.6.2, 11.6.3, 11.6.4, 11.6.5, 12.1.1, 12.1.2, 12.2.2, 12.2.3, 12.2.4, 13.2, 13.3 from EN 301 549"Accessibility requirements suitable for public procurement of ICT products and services in Europe", available at http://www.etsi.org/deliver/etsi_en/301500_301599/301549/01.01.01_60/en_301549v010101p.pdf

The supplier may answer such requirement by third party certification or a declaration of conformity. On one hand the risk of such procurement policies is the risk of under-specification by not requiring supporting a whole standard. On the other hand it will lead to a quite long list of individual requirements like the above that makes it difficult for small vendors to answer such tenders. Nonetheless such practice is becoming common throughout Europe and it is viable to ask if such modular requirement lists can be referenced by modular components.

3.1.6.1 Use within the Developer Space

Based on the in depth analysis by the Standardization Task we believe that the Developer Space can actually contribute to the fulfilment of this Standard. Building upon the idea of the mandate376 website, we can provide an according categorization scheme based on particularly section 9ff of the standard. By allowing facetted searches based on the categorization scheme of the standard we help companies meeting the standard in places their current product has deficits. Particularly combining this search mechanism by a technology oriented filtering, companies may quickly find fitting open source technologies that enable them to reply to tenders.

Based on preliminary categorization of the Developer Space prototype by the standardization task within Prosperity4ALL we have implemented a mechanism to do categorization of the main site.

As components and resources seldom lead to complete fulfilment of a technical requirement, we have added the possibility to tag the level of support as well give a comment. The work on categorization is not yet completed. The full categorization scheme will be presented in the final deliverable D203.3, which will describe the complete live Developer Space site.

4 Platform Level Accessibility

4.1 Interoperability between Information Technology (IT) and Assistive Technology (AT) (ISO/IEC 13066)

While accessibility has become an essential part in many operating systems and is mainly driven by or through the vendors, ISO/IEC 13066 has set out to define interoperability with Assistive Technology on a platform level. The first part 13066-1 is a cross-platform standard for designing and evaluating interoperability between IT and AT regarding three types of interoperability: hardware-to-hardware, hardware-to-software, and software-to-software.

Part 2 of this standard defines, among other things, the Windows Accessibility Application Programming Interface (API) that was de facto standardized before via the Microsoft Windows Automation Frameworks, including Microsoft Active Accessibility, User Interface (UI) Automation, and the common interfaces of these accessibility frameworks including the IAccessibleEx interface specification. This standardization aims only at one operating system. Still, it provides stability particularly towards AT Vendors as it formally specifies services provided in the Microsoft Windows platform to enable AT to interact with other software within Windows. Products are most reliable when standardized public interfaces are used. It describes requirements for IT and AT products in three types of interoperability: hardware-to-hardware, hardware-to-software, and software-to-software. It also identifies a variety of software-to-software APIs that are described further in other parts of ISO/IEC 13066. These APIs can be used as frameworks to support IT-AT interoperability.

In its third part, the standard defines IAccessible2, an extension of the Microsoft Active Accessibility API designed by IBM and now an open standard. IAccessible2 constitutes a cross-platform approach to desktop accessibility, as an extension of the Microsoft Active Accessibility API designed by IBM. It bridges the gap towards Accessible2 fills in perceived omissions in MSAA to match the Java Accessibility API and Assistive Technology Service Provider Interface (AT-SPI) commonly used on Linux Desktop Systems.

4.1.1.1 Use within the Developer Space

It can be seen as a trend that less components directly support AT interop interfaces. Within the Developer Space most rely on frameworks to bridge to this technology. Lately with the the rise UI Frameworks like Electron or the Universal Windows Platform, and the general move to Web Application the importance of those interfaces in the development community shrinks which is reflected also in the components listed in Developer Space. Most importantly browsers or GUI-Frameworks like Gnome implement those interoperability standards correctly to eg. work with screen readers, which are in turn addressed by components.

4.2 Accessibility in Mobile Platforms

As mentioned above, all major smartphone platforms provide support and guidelines for development of accessible technology. It should be noted that these are only de facto

standards and they are only encouraged, but not mandatory for publication of applications. Even the strict reviews of Apples AppStore do not check for accessibility. All platforms have checklists as well as testing tools and frameworks to design for accessibility.

Although having only a minimal market share, Microsoft already has extensive support for AT in their mobile operating system and guides developers how to design accessible applications. This stems from the fact that they encourage the development of unified applications, which run on mobile and desktop devices alike. Therefore, they are building on their extensive experience described above. Apple also transferred their knowledge from the desktop platform, although different mechanisms and guidelines are imposed for both systems. The main focus of the iOS guidelines, similar to the other manufacturers, is the compatibility with the text-to-speech function, which is provided system-wide. API functions are therefore focused on providing captions and labels, but there are also recent additions to specify custom actions for accessible interfaces. Android is very open for enhancements and provides the possibility to register background services that handle different input and output-modalities. Therefore a user is able to customize his interface through installation of matching services via the app store, given that the used applications include the necessary annotations.

A special case are cross-platform applications, which are built with Web technologies and are executed in a browser view on each of the operating systems. These are usually created with frameworks like Apache Cordova, which generated the application packages for the different platforms. On one hand, they already support the accessibility technologies of the browser view, used for rendering and profit from the efforts made for Web technologies. On the other hand, they are detached from the possibilities provided by the base operating system and do not benefit directly from neither the optimizations of native UI elements, nor tools and mechanisms provided by the manufacturer. Therefore, additional plugins are needed to access the platform specific APIs.

4.2.1.1 Use within the Developer Space

Similar as other applications, the importance of Web Accessibility (see next section) grows also in the mobile domain. Packages like the phone gap-mobile-accessibility help exposing native APIs to web based application. Only few open source accessibility components particularly address explicitly mobile platforms. Examples are the EVA Facial Mouse for Android or the Android Vibration Module developed by CERTH. Also only comparably few testing tools exist like Louis for iOS or Google's Accessibility Test Framework for Android both not receiving all to much developer attention compared to their web counterparts.

5 Web Accessibility

Given the GPIIs focus on Internet technology, a main focus of standardization is Internet technology. Particularly standardization regarding Application, Presentation and Session protocols (OSI layer 5-7) are important in the scope of the GPII. In the internet this technology is summarized by the World Wide Web, and mostly standardized by the W3C (in contrast to the IETF for lower networking protocols). Particularly APIs supported by Web Browsers are standardized by the W3C. The most notable standards are HTML, XML, CSS and DOM.

However, the most common scripting language is standardized by the ECMA (ECMAScript aka JavaScript). These standards, however, are typically not competing but referencing each other. JavaScript was established as a de facto standard by Netscape and contributed to the ECMA for standardization under ECMA-262. JavaScript is a good example of the varying scope of standardization (ECMA does not standardize all extensions that are commonly supported by JavaScript). It is also an example where standardization has led to competing standards, i.e. JavaScript Object Notation (JSON, ECMA-404, RFC 7159) that is commonly used in many components of the GPII. It is good for developers to be aware of this. However, practical cross-browser support (de facto standardization) and actively tested interoperability is far more relevant.

The most important standard that is shaping the Web is HTML (that primarily describes the structuring and presentation of content for web browser). An important development with the final standardization of HTML5 in 2014 is that many technologies that were part of the original HTML5 standardization process were moved to separate specifications (often maintained outside the W3C):

- HTML Canvas 2D Context
- Web Messaging,
- Web Workers,
- Web Storage,
- WebSocket API,
- Server-Sent Events,
- WebSocket Protocol,
- WebRTC
- WebVTT

Other extension that are referenced by the HTML standard most notably include:

- SVG
- MathML
- WAI-ARIA

All those web standards are important for the Developer Space as a common denominator. These extensions particularly offer many innovative uses of Web Technology for assistive technology. It is, however, clear that even this set of standards is very wide and only defines APIs and mark-up that can be used by web application that can run across multiple vendors.

Particularly those specifications do not guarantee interoperability of different components and compatibility with commonly used assistive technology out of the box. Lately the scope of HTML5 even broadened as one of the most promising frameworks for cross-platform for mobile applications.

As Web standards have been designed with accessibility in mind, the extension of the scope of Web standards towards mobile application is a huge opportunity for tackling accessibility barriers in many domains beyond the stationary Internet. Thus a focus of this document is dedicated towards relevant web Standards as a potential for future assistive technology and inclusive component design.

5.1.1 Web Content Accessibility Guidelines (WCAG)

One of the major milestones in IT Accessibility were the Web Content Accessibility Guidelines. The first version was published by Gregg Vanderheiden just after accessibility was raised in the second World-Wide Web by Tim Berners-Lee.

The current version 2 of Web Content Accessibility Guidelines aim at making Web content more accessible for people with all kinds of disabilities, but also for elderly people. The goal of WCAG 2.0 is providing a single shared standard for web content accessibility that meets the needs of individuals, organizations, and governments internationally.

WCAG is primarily intended for:

- Web content developers (page authors, site designers, etc.)
- Web authoring tool developers
- Web accessibility evaluation tool developers
- Others who want or need a standard for web accessibility

Thus, the WCAG as a fundamental guideline plays an important role particularly for the web components and media transformation components, but is also applicable throughout the Developer Space (particularly the very own interface of the Developer Space listing).

At the top are four principles that provide the foundation for Web accessibility: perceivable, operable, understandable, and robust.

Under the principles are guidelines. 12 guidelines provide the basic goals that authors should work toward in order to make content more accessible to users with different disabilities. For each guideline, testable success criteria are provided to allow WCAG 2.0 to be used where requirements and conformance testing are necessary such as in design specifications, purchasing, regulation, and contractual agreements. According to those guidelines three levels (A, AA, AAA) of accessibility conformance can be reached.

The standard is explained for adopters in multiple supporting Documents:

1. How to Meet WCAG 2.0 - A customizable quick reference to WCAG 2.0 that includes all of the guidelines, success criteria, and techniques for authors to use as they are developing and evaluating Web content.

2. Understanding WCAG 2.0- A guide to understanding and implementing WCAG 2.0. There is a short "understanding" document for each guideline and success criterion in WCAG 2.0 as well as key topics.

3. Techniques for WCAG 2.0- A collection of techniques and common failures, each in a separate document that includes a description, examples, code and tests.

(Source: http://www.w3.org/WAI/intro/wcag.php)

There is a diagram and description of how the technical documents are related and linked. Further educational materials available via the working group homepage within the W3C.

5.1.1.1 Use within the Developer Space

More than 60 components in the Developer Space refer to the WCAG standard. Many of those components provide automated testing for the WCAG standard in general, or specific elements like the Photosensitive Epilepsy Analysis Tool by TRACE. Other tools help to fulfill e.g. the contrast requirements named in the standard.

5.1.2 Accessible Rich Internet Applications Suite (WAI-ARIA)

While HTML generally should be accessible, WAI-ARIA helps with rich internet applications using dynamic content and advanced user interface controls. It is currently ranked as an official extension to HTML5. Especially it focusses on providing web developers the ability to better support people who rely on screen readers and people who cannot use a mouse. Particularly it does so by defining:

- Roles
  - to describe the type of widget presented, such as "menu", "treeitem", "slider", and "progressmeter"
  - to describe the structure of the Web page, such as headings, regions, and tables
- Properties
  - to describe the state that widgets like a check box are in
  - to define dynamic regions of a page that are likely to get updates
  - drag-and-drop that describe drag sources and drop targets
- A way to provide keyboard navigation for the Web objects and events, such as those mentioned above.

Commonly a lot of HTML5 based user interface frameworks use WAI-ARIA as an interface for providing advanced accessibility. Often this is still done via special templates or plugins or special configuration is needed. The Web Accessibility Component section links a few of those plugins or guides. The Mozilla Developer Networks also give a lot of useful links regarding WAI-ARIA support.

5.1.2.1 Use within the Developer Space

More than 100 components and tools on the Developer Space refer the ARIA specification. Most of these components make it easier to add correct semantic ARIA mark-up to web development frameworks like react. Components range from Video Player to general Modal Windows. Also developer tools help to write correct ARIA compliant HTML mark-up or provide specific tests. ARIA compliance is the by far largest area of components.

5.1.3 Remote Control

The ISO/IEC 24752 standards specify the basic concepts and document formats for the Universal Remote Console ecosystem. In its second edition, the URC technology has been revised based on lessons learned to be simpler to implement and based on current technologies. The URC technology is part of the GPII concepts. Through its separation of frontend and backend, it allows for pluggable user interfaces that may be provided by the manufacturer of a device or by third parties. In the AAL context, URC is a basic technological concept for auto-personalization by preferences that particularly targets control of devices and services.

URC particularly provides for abstracting different target protocols. The new part 6 of ISO/IEC 24752 defines a way for devices/services to expose themselves as SOAP-based Web services in a URC-compliant way. The Universal Control Hub can be construed as a proxy target that provides access to other targets. The URC-HTTP protocol of the UCH and the URC-HTTP Target protocol is predecessors for a new RESTful protocol for targets to be standardized (see the GPII Standardization Roadmap).

Device templates are strongly needed for interoperability of pluggable user interfaces across device models. Device manufacturers can use the appropriate templates as a basis for their products. With the new inheritance feature (see ISO/IEC 24752-2:2014) user interface sockets can be easily extended for product-specific additions. However, the pluggable user interfaces for the device template can still be used for the extended product (albeit hiding the additional features). A set of device templates consists of the following documents:

- Target description template
- Socket description template
- Grouping sheet template
- Resource sheet template
- Optional: WSDL1 and WSDL2 documents

One contribution of the Prosperity4All http://ds.gpii.net are so called URC Super Sockets that will work for multiple targets and thus lower the barriers for adapting this technology for new devices.

5.1.3.1 Use within the Developer Space

Looking at the Developer Space, the development of URC specific components has stalled particularly outside the Prosperity4All project. Yet we see high potential for adoption of

newer parts of the standard. Within the Developer Space OpenAPE e.g. provides a reference implementation of ISO/IEC 24752-8, that features lightweight adaption. While Amazon Echo, Google Home and Apple HomeKit (see above on de-facto standardization) dominate the market for remote controlling applications in the Smart Home, the Eclipse Smart Home Project provides an basis for open development. Professional vendors like the Deutsche Telekom support those developments by adapting the Eclipse Smart Home based QIVICON platform as basis for their Smart Home platform. While work on providing socket templates for the URC middleware (ie. for use with the Universal Control Hub) has not gained wide adoption, the concepts and work can be transported easily to the abstractions of the Eclipse Smart Home (i.e. abstractions like thing types and channel types).1. Yet adoption by accessibility related components is yet to be seen. Many accessibility related components like the Smart Home Examples from the AsTeRICS project still rely on direct support for home automations standards like KNX,enOcean or FS20 with no intermediated abstractions in between.

5.1.4 Haptic and Touch Interfaces

One specific part of ISO 9241 "Ergonomics of human-system interaction" is the specification of haptic interfaces. ISO 9241-910 provides a framework for understanding and communicating various aspects of tactile/haptic interaction. It defines terms, describes structures and models, and gives explanations related to the other parts of this ISO 9241 "900" subseries. It also provides guidance on how various forms of interaction can be applied to a variety of user tasks. It is applicable to all types of interactive systems making use of tactile/haptic devices and interactions. It does not address purely kinaesthetic interactions, such as gestures, although it might be useful for understanding such interactions.

Part 920 of ISO 9241 gives recommendations for tactile and haptic hardware and software interactions. It provides guidance on the design and evaluation of hardware, software, and combinations of hardware and software interactions, including: the design/use of tactile/haptic inputs, outputs, and/or combinations of inputs and outputs, with general guidance on their design/use as well as on designing/using combinations of tactile and haptic interactions for use in combination with other modalities or as the exclusive mode of interaction; the tactile/haptic encoding of information, including textual data, graphical data and controls; the design of tactile/haptic objects, the layout of tactile/haptic space; interaction techniques.

It does not provide recommendations specific to Braille, but can apply to interactions that make use of Braille. The recommendations given in ISO 9241-920:2009 are applicable to at least the controls of a virtual workspace, but they can also be applied to an entire virtual environment — consistent, in as far as possible, with the simulation requirements. (NOTE: It is recognized that some interactive scenarios might be constrained by the limitation that a real workspace is to be modelled in a virtual environment. Objects can be in suboptimal positions or conditions for haptic interaction by virtue of the situation being modelled)

ISO 9241-940 covers the evaluation of tactile / haptic interactions

Few standardized APIs for haptic feedback exist. The W3C Vibration API is specifically designed to address use cases that require simple tactile feedback only. Use cases requiring more fine-grained control are out of scope for this specification. This API is not meant to be used as a generic notification mechanism. Such use cases may be handled using the Notifications API specification. In addition, determining whether vibration is enabled is out of scope for this specification.

CHAI3D is an open source set of C++ libraries for computer haptics, visualization and interactive real-time simulation. CHAI 3D supports several commercially-available three-, six- and seven-degree-of-freedom haptic devices, and makes it simple to support new custom force feedback devices. CHAI 3D is especially suitable for education and research purposes, offering a light platform on which extensions can be developed. CHAI 3D's support for multiple haptic devices also makes it easy to send your applications to remote sites that may use different hardware.

In short, CHAI 3D takes an important step toward developer-friendly creation of multimodal virtual worlds, by tightly integrating the haptic and visual representations of objects and by abstracting away the complexities of individual haptic devices.

5.1.4.1 Use within the Developer Space

Currently the AndroidVibrationModule and the WebHapticAPI developed within the Prosperity4All project are the only components in the Developer Space that target haptic interaction. In terms of standards and interfaces, the AndroidVibrationModule hooks against the default Android events API. Modules do not explicitly list any relations to Standards. However defacto Standard APIs like the CHAI3D API is e.g. bridged to the web development world by the WebHapticModule.

5.1.5 Multimodal Interfaces

ETSI EG 202 048 presents guidelines for the design and use of multimodal symbols using a Design for All approach. It also provides a study of the needs and requirements for the use of multimodal symbols in user interfaces,with special emphasis on the requirements of people with disabilities and elderly people. ETSI EG 202 048 provides guidelines, good practice and case studies for the successful design and application of multimodal symbols using the Design for All approach. It will support the standardization process with respect to the use of multimodal symbols in modern user interfaces. Icons, symbols and pictograms are widely used components of user interfaces in ICT applications and services, e.g. for navigation, status indication and function invocation. Examples of such applications and services include information retrieval (e.g. Web sites), messaging (e.g. email and SMS), public services (e.g. public telephones and ATMs) and real time communication services (e.g. fixed and mobile telephony). The use of visual-only symbols in such applications and services creates temporary or permanent problems for all users. User groups most affected are blind and partially sighted people and users of mobile devices with limited visual display capabilities. All users can potentially benefit from the current and future possibilities of multimodal user interfaces. These interfaces combine communication channels, for example sound, graphics, video, speech, force and vibration. ETSI EG 202 048 does not deal with unimodal symbols,

i.e. only visual symbols or only auditory "earcons", but with symbols that use at least two communication channels.

5.1.5.1 Use within the Developer Space

Currently only ARIA (and thus screen reader support) for web based symbol collections like font-awesome or SVG symbols can be found in the Developer Space. Components for true multimodal symbols are still not available in our open source collection.

6 Related and Future Work

One challenging part apart from establishing standards and APIs was establish to quality standards for the components. Within user evaluations particularly this aspect was highlighted from professional developers in the AT domain. Based on work on the Software Sustainability Maturity Model 2 and on other curation efforts 3 we have defined four levels of software maturity that we can be used in self-rating approach in the Developer Space:

- Demonstrated Reuse
- Highly reusable
- Reusable
- Basically Reusable
- Reuse with risk

We have added simple criteria to each level based on which we are currently asking all component developers to self-rate their components. As this is difficult to sustainably provide to the vast number of more 800 components included or queued for inclusion the Developer Space we are still looking into different aspects that can be measured automatically to provide such similar metrics. The number of stars (ie. likes) and the number of forks (modifications of the software) on github have proven to be interesting metrics as well as the number of commits and resolved issues as reported by the Quality Infrastructure chart components. User rating and links to the actual use of software might also provide valuable information. We furthermore decided to crawl up-to-date information on GitHub instead of editing custom descriptions as they display much better the maturity of a component that edited content.

As with other standards, we have learned that it does not actually increase adoptions if developers are forced into standard convergence. Particularly the work on an integrated runtime has shown, that fully integrated solutions also add complexity. The research work in the Prosperity4ALL project showed particularly the value in providing lightweight loosely coupled solutions with a low requirement profile. Rather stand-alone solutions like the AsTeRICS CameraMouse or the UI Options panel were among the most adopted components as they require low overhead in provisioning while providing visible added value to products. Particularly in the web development domain technology choices were mostly influenced by concrete frameworks and interoperability concerns beyond standards. Thus it was most valuable here to provide multiple comparable choices e.g. for accessible charting components, rather than to provide full-blown frameworks. Technological decisions are still mostly made with not primarily accessibility in mind, however particularly with public procurement regulations in place we see a stronger interest in those factors. We have reacted particularly to that need by adding categorizations based on “EN 301 549” (which in turn references e.g. WCAG) into our Developer Space.

Those standards ensure basic accessibility. For advanced practical accessibility challenges and new AT products, we see particularly a need for inclusive technology to adapt to commonly used technologies. While reactive web applications or augmented reality has opened a full range of possibilities, the technologies are constantly evolving. Frameworks are changing constantly and solutions will be measured on how well they can follow

technological trends. Only core standards like HTML5 can establish basic accessibility at relatively low maintenance cost. Beyond that broad inclusion can be only delivered through standards that become part of regulation. As this mostly covers only functional requirements, vendors still need to be able to adapt rapidly to changing interoperability technologies. A component listing like in the Developer Space can help in this effort by providing a broad range of technologies to choose from and by providing an entry point to an open source community that helps to spread the cost of constant change.