Barrierefreiheit von Web Anwendungen mit dem ARIA Standard sicherstellen

Erste Erfahrungen aus einem Anwendungsprojekt

Annett Hardt
SAP AG – User Experience
Dietmar-Hopp-Allee 16
69190 Walldorf, Germany
annett.hardt@sap.com

Martin Schrepp
SAP AG – User Experience
Raiffeisenring 45
68789 St. Leon-Rot
martin.schrepp@sap.com

Abstract
Das Internet hat sich von einem reinen Informationsmedium zu einem Zugang für viele wichtige Dienstleistungen entwickelt. Für die Teilnahme behinderter Menschen am gesellschaftlichen Leben ist die Barrierefreiheit solcher Online-Dienstleistungen eine wichtige Voraussetzung. Allerdings handelt es sich bei vielen dieser Angebote nicht mehr um einfache HTML-Formulare, sondern um komplexe interaktive Anwendungen. Solche interaktiven Web-Anwendungen barrierefrei zu gestalten, kann mit den herkömmlichen Mitteln schwierig sein. Hier soll der neue ARIA (Accessible Rich Internet Applications) Standard des W3C Abhilfe schaffen. Die Grundidee des ARIA Standards ist es, vorhandene Markup-Sprachen anzureichern, um assistiven Technologien die notwendigen Informationen zu Art, Zustand und Veränderungen von Elementen der Benutzeroberfläche zu liefern. Wir beschreiben die Grundprinzipen hinter dem ARIA Standard anhand praktischer Beispiele aus einem Projekt. Dabei werden wir die Vorteile dieses neuen Standards herausarbeiten, aber auch auf nach wie vor bestehende Probleme hinweisen. Es wird weiterhin über konkrete Erfahrungen zur Umsetzung des Standards im Projekt berichtet.

Keywords:
/// Barrierefreiheit
/// ARIA Standard
/// interaktive Web-Anwendungen

1. Einleitung

Das Internet hat sich in den letzten Jahren von einem reinen Informationsmedium zu einem Zugang für viele wichtige Dienstleistungen entwickelt. Hier fallen einem natürlich zuerst eCommerce Lösungen ein, die es erlauben eigentlich alle Arten von Produkten direkt in einem Online-Shop zu bestellen, den Ablauf der Auslieferung zu verfolgen und fehlerhafte Lieferungen zu reklamieren. Aber auch im öffentlichen Bereich (eGovernance) werden zunehmend Leistungen über Online-Services angeboten. Beispiele sind das Beantragen eines neuen Personalausweises, Meldungen zu Adressänderungen innerhalb einer Gemeinde, das Beantragen behördlicher Dokumente (z. B. eines Gewerbescheins) oder die Abgabe der Steuererklärung. Auch viele kommerzielle Dienstleister bieten mittlerweile ihren Kunden Online-Dienste zur Vereinfachung von Verwaltungsvorgängen an. Ein Beispiel ist hier die von vielen Energieversorgungsunternehmen angebotene Möglichkeit, Zählerstände für Wasser oder Stromverbrauch selbst abzulesen und über das Internet zurückzumelden.

Für die möglichst uneingeschränkte Teilhabe behinderter Menschen am gesellschaftlichen Leben ist die Barrierefreiheit solcher Angebote eine sehr wichtige Voraussetzung. Das Design der Benutzungsschnittstelle solcher Anwendungen sollte sich daher am Prinzip des Universellen Designs (siehe z. B. Stephanidis & Salvendy, 1999 oder Bühler & Stephanidis, 2004) orientieren. Ziel dieses Designkonzepts ist es, Benutzungsoberflächen so zu gestalten, dass diese von möglichst vielen Menschen direkt oder mit Hilfe assistiver Technologien (z. B. Screen Reader, Braillezeile oder Screen Magnifier) genutzt werden können.

Allerdings handelt es sich bei solchen Online-Diensten oft nicht mehr um einfache HTML-Formulare oder statische Web-Seiten, sondern um zum Teil recht komplexe interaktive Anwendungen. Diese basieren in der Regel auf HTML in Verbindung mit Java-Script oder anderen Technologien, wie z. B. MS Silverlight oder Adobe Flex.

Solche interaktiven Web-Anwendungen barrierefrei zu gestalten, kann mit den herkömmlichen Mitteln schwierig sein. Hier soll der neue ARIA (Accessible Rich Internet Applications) Standard des W3C Abhilfe schaffen. Dieser Standard ist im Moment im Zustand Candidate Recommendation, d.h. dieser Standard ist noch nicht endgültig verabschiedet. Allerdings haben die Anbieter von Browsern oder assistiven Technologien, wie z. B. Screen Readern, schon begonnen den Standard zu unterstützen. Es ist damit möglich diesen Standard bereits jetzt in Projekten einzusetzen, um die Barrierefreiheit hoch interaktiver Benutzungsoberflächen im Internet sicherzustellen. Wir beschreiben in diesem Beitrag den Einsatz des ARIA Standards in einem Projekt der SAP AG.

2. Wie funktioniert der ARIA Standard?

Die Grundidee des ARIA Standards (Craig & Cooper, 2011) ist es, vorhandene Markup-Sprachen anzureichern, um assistiven Technologien die notwendigen Informationen zu Art und Zustand von Elementen der Benutzeroberfläche zu liefern. Vorhandene Markup-Sprachen, wie z. B. HTML, stellen schon eine Reihe von Mechanismen zur Herstellung von Barrierefreiheit zur Verfügung. Ziel des ARIA Standards ist es nicht, diese Mechanismen komplett zu ersetzen, sondern die Lücken in den vorhandenen Markup-Sprachen zu füllen. D.h. die Vorgabe des Standards ist es, nur dann auf die ARIA Mechanismen zurückzugreifen, wenn keine entsprechende Möglichkeit in der verwendeten Technologie vorhanden ist.

Zum Beispiel bietet HTML die Standardmöglichkeit ein Eingabefeld mit einem Bezeichner zu versehen (über <label for="id" …>Labeltext</label>). Auch der ARIA Standard definiert eine solche Möglichkeit (über die Eigenschaft aria-labelledby). In einer auf HTML basierenden Anwendung sollte man aber immer den Standardmechanismus, d.h. in diesem Fall das Label-Tag verwenden, bevor man auf die entsprechenden ARIA Mechanismen zugreift.

Der ARIA Standard definiert Rollen (roles), Eigenschaften (properties) und Zustände (states):

- Eine Rolle beschreibt den Typ eines Elements der Benutzerschnittstelle. So gibt es zum Beispiel die Rollen Checkbox, Button und Slider.
- Eigenschaften sind Attribute, die sich auf eine Rolle beziehen und eine dauerhafte Gültigkeit besitzen. Beispiele sind die Eigenschaften aria-labelledby (Identifiziert den Bezeichner eines Elementes) oder aria-autocomplete (zeigt an, ob für das Element Eingabevorschläge vorhanden sind).
- Für die Beschreibung von Attributen, die eine eher kurzfristige Gültigkeit haben, werden Zustände verwendet. Beispiele für solche Zustände sind aria-checked (gibt für eine Checkbox oder einen Radiobutton an, ob er im Moment markiert ist oder nicht) oder aria-invalid (kann verwendet werden, um anzuzeigen, dass der in ein Control eingegebene Wert nicht korrekt ist).

Zustände können sich also während einer Interaktion eines Nutzers häufiger ändern, Eigenschaften bleiben im Normalfall unverändert.

Die Informationen zu den Rollen und Eigenschaften können in der Regel statisch zum Markup eines Controls hinzugefügt werden. Die Änderung der Zustände eines Controls muss abhängig von den Nutzeraktionen über Scripting angepasst werden. Zum Beispiel muss beim Klicken auf eine bisher unmarkierte Checkbox deren Zustand per Scripting von aria-checked = false auf aria-checked = true geändert werden.

3. Vorteile des ARIA Standards

Der Control-Umfang der meisten Markup-Sprachen ist eher auf statische Webseiten ausgelegt. Bei sehr interaktiven Web-Anwendungen stößt man hier schnell an Grenzen. Zum Beispiel ist in HTML kein Slider-Control vorhanden. Der ARIA Standard bietet über die vorhandenen Rollen die Möglichkeit auch Controls zu nutzen, die in der verwendeten Markup-Sprache nicht vorhanden sind.

Bisher wurden solche nicht vorhandenen Controls in den Markup-Sprachen durch Kombination vorhandener Gestaltungselemente und den massiven Einsatz von Scripting nachgebaut. Für blinde Nutzer ist eine solche Vorgehensweise sehr kritisch, da hier eine Menge von Controls ineinander geschachtelt wird und dabei ihre eigentliche Semantik verloren geht. Ein Screen Reader wird einem blinden Anwender einfach all diese geschachtelten Elemente nacheinander vorlesen, was in diesem Fall nicht sinnvoll ist. Einem solchen „nachgebauten" Control kann mit Hilfe des ARIA Standards seine korrekte Rolle zugewiesen werden. Ein Screen Reader ist damit in der Lage das semantisch richtige Control zu erkennen und dem Benutzer dessen Bezeichnung mitzuteilen. Hilfselemente, die nur für die visuelle Darstellung benötigt werden, können mit Hilfe der Rolle Presentation als semantisch unwichtig deklariert werden. Solche Elemente werden dann bei der Ausgabe über einen Screen Reader nicht vorgelesen.

Ein weiteres Problem war bisher, dass einige vorhandene Controls in HTML nur bedingt visuell veränderbar sind. Dadurch ist es unter Verwendung der Standard-Controls oft nur schwer möglich, ein in sich stimmiges visuelles Design zu erhalten. Daher werden in HTML vorhandene Controls manchmal aus Gründen der visuellen Gestaltung durch eine Kombination anderer HTML Elemente simuliert. Dies zieht aber unmittelbar Probleme für Nutzer assistiver Technologien nach sich, da die assistive Technologie in solchen Fällen nicht in der Lage ist, dem Benutzer die korrekte Semantik mitzuteilen.

Betrachten wir dies an einem Beispiel. Der HTML Button besitzt gewisse Einschränkungen in Bezug auf die visuelle Gestaltung. Daher wird ein Button auf der Nutzungsoberfläche oft technisch als speziell gestylter HTML Link realisiert. Ein sehender Anwender wird dies bei geeigneter visueller Gestaltung nicht bemerken, da der Link auf der Oberfläche für ihn wie ein Button aussieht. Blinde Anwender, die auf die Verwendung eines Screen Readers angewiesen sind, können hier aber Probleme bekommen, da der Screen Reader korrekt einen Link erkennt und dies dem Benutzer auch so vorliest.

Mit Hilfe von ARIA kann dieses Problem sehr elegant gelöst werden. Man ergänzt einfach das Markup des Links mit der ARIA Rolle Button. Wenn ein Screen Reader die entsprechende Markup-Information ausliest, bekommt die ARIA Rolle eine höhere Priorität als die technische Information in HTML und somit wird dem blinden Anwender nun semantisch korrekt mitgeteilt, dass es sich hier um einen Button handelt. Die rein technische Markup-Information wird vom Screen Reader komplett ignoriert.

Code für einen über einen Link simulierten Button:

```
<a id="B1" title="Button"
href="…"> … </a>
```

Code mit Verwendung des ARIA Standards:

```
<a role="button" id="B1"
title="Button" href="…"> … </a>
```

Ein weiteres Beispiel, bei dem Einschränkungen bzgl. der visuellen Gestaltungsmöglichkeiten häufig zum Nachbau des vorhandenen HTML Controls führen, sind Checkboxen. Um ein ansprechendes oder originelles Layout zu erzielen, wird eine Checkbox häufig durch die Kombination eines Links mit zwei Grafiken (diese visualisieren die Zustände Markiert und Nicht markiert) realisiert, z. B. wenn die Checkbox abgerundete Ecken haben soll.

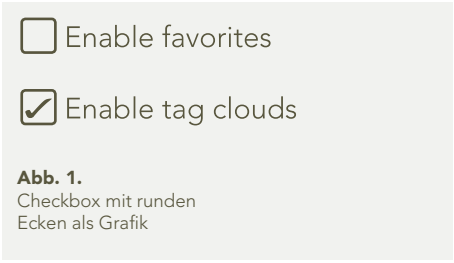

Abb. 1.
Checkbox mit runden Ecken als Grafik

Klickt der Nutzer auf die Checkbox, so werden diese Grafiken über Java-Script ausgetauscht. Ein Screen Reader wird dem Anwender mitteilen, dass es sich bei dem Control um einen Link mit Bild handelt. In diesem Beispiel kann ein blinder Anwender nicht erkennen, dass hier eigentlich eine Checkbox realisiert ist. Bisher konnte man solche Probleme nur über zusätzliche Textinformationen lösen, z. B. indem man über den Tooltip mitgeteilt hat, dass es sich um eine Checkbox handelt. Eine alternative Lösungsmöglichkeit bestand darin, einen speziellen Modus für blinde Nutzer anzubieten, in dem ausschließlich die nativen HTML Controls verwendet werden. Mit dem ARIA Standard kann dies nun über die entsprechende Rolle korrigiert werden. Der Link muss dazu die Rolle Checkbox erhalten, das Bild erhält die Rolle Presentation (wodurch es vom Screen Reader ignoriert wird). Das vorhandene Java-Script zum Tauschen der Bilder muss zusätzlich den Zustand zwischen aria-checked = true und aria-checked = false hin und herschalten.

Code für eine über zwei Bilder simulierte Checkbox:

```
<a id="CB1" title="Checkbox"
onclick="Java-Script zum Aus-
tauschen des Bildes">
<img src="uncheckedcb.jpg" />
</a>
```

Code mit Verwendung des ARIA Standards:

```
<a role="checkbox"
aria-checked="false" id="CB2"
onclick=" JavaScript zum Aus-
tauschen des Bildes und
Umschalten des Zustandes">
<img role="presentation"
src="uncheckedcb.jpg" />
</a>
```

Bisher war es in komplexen und stark interaktiven Web-Anwendungen immer kritisch, Fehlermeldungen für einen blinden Anwender zugänglich zu machen. In der Regel wird hierfür ein fester Bereich auf dem Bildschirm definiert, der alle Fehler, Warnungen und Erfolgsmeldungen anzeigt. Selbst wenn ein blinder Anwender weiß, dass ein solcher Bereich existiert, muss er ständig prüfen, ob dort gerade Fehlermeldungen angezeigt werden. Auch hierfür bietet der ARIA Standard eine elegante Lösung an. Mit ARIA ist es möglich eine sogenannte Live-Region zu definieren. Eine Live-Region ist ein Bereich, dessen Inhalt sich häufig ändert und bei dem die Veränderungen dem Benutzer mitgeteilt werden sollen. Ein Screen Reader kann damit automatisch Änderungen in dieser Region verfolgen und diese dem Anwender vorlesen. Für eine solche Live-Region kann man definieren, ob neue Inhalte immer sofort oder erst zu einem sinnvollen Zeitpunkt (z. B. am Ende eines Satzes) vorgelesen werden sollen und ob immer der komplette Inhalt oder nur der neue bzw. veränderte Inhalt vorgelesen werden soll.

4. Noch bestehende Probleme

Trotz der vielen Verbesserungen und Erleichterungen, die der ARIA Standard mit sich bringt, gibt es noch einige Bereiche, die noch nicht völlig befriedigend abgedeckt sind.

Zum Beispiel bietet der ARIA Standard keine Schnittstelle, um eigene Rollen definieren zu können. Dies ist für einfachere Web-Anwendungen sicher kein grosses Problem, da der Standard schon einen ausreichend grossen Umfang an Rollen definiert.

Schwierig kann dieser Punkt insbesondere bei komplexeren betriebswirtschaftlichen Web-Anwendungen werden (siehe z. B. Jani & Schrepp, 2005 oder Hardt & Schrepp, 2007). Hier werden oft auf spezielle Nutzungsszenarien zugeschnittene komplexe Controls definiert, die dann an vielen Stellen einer Anwendung verwendet werden (z. B. Gantt-Charts zur Darstellung von geplanten Zeiträumen oder graphische Editoren zur Darstellung und Veränderung von Objektbeziehungen). Der Bedarf geht hier deutlich über die im Standard vorhandenen Controls hinaus. Für solche Szenarien wäre es wichtig, eigene Rollen definieren zu können, um die Barrierefreiheit dieser Controls dann ebenfalls mit Hilfe des ARIA Standards sicherstellen zu können.

Ein weiteres potentielles Problem kann durch die schnelle Entwicklung neuer Controls entstehen, die dann oft sehr schnell Verbreitung finden. Da sich ein internationaler Standard wegen des hohen Abstimmungsbedarfs natürlich nur langsam weiter entwickeln kann, besteht die Gefahr, dass neue populäre Controls erst mit großer Verzögerung in eine neue Versionen des ARIA Standards aufgenommen werden können.

Ein weiteres Problem ist es, dass Textformatierungen nicht ausgedrückt werden können. Beispiele sind rote (z. B. um einen negativen Betrag hervorzuheben) und grüne (um einen positiven Betrag

hervorzuheben) Zahlen in einer Tabelle oder durchgestrichene Beträge in einem Web-Shop (die den alten nicht mehr gültigen Preis ausdrücken). Solche Formatierungen haben im Kontext einer Seite oft eine eindeutige Bedeutung und helfen einem sehenden Anwender sich zu orientieren. Die meisten assistiven Technologien unterstützen die Bekanntgabe dieser Formate nicht, somit kann diese Information einem blinden Anwender nicht mitgeteilt werden. Hier ist es auch zukünftig notwendig, dass solche Formatierungen dem blinden Anwender über textuelle Zusatzinformationen, wie z. B. einem Tooltip zur Kenntnis gebracht werden.

Ein weiteres offenes Thema ist die Verwendung von mehreren Rollen gleichzeitig. Stand heute ist es immer nur möglich genau eine Rolle für ein Control zu definieren. In Web-Anwendung gibt es aber zum Beispiel häufig den Fall, dass eine Überschrift gleichzeitig ein klickbarer Link ist. In diesem Fall muss sich ein Entwickler entscheiden, welche Rolle (die Überschrift für die Orientierung oder der Link für die Interaktion) für den blinden Anwender wichtiger ist. Dieser Punkt ist allerdings nur dann kritisch, wenn dieses Element nicht mit dem dafür vorgesehen HTML Standardelementen (z. B. <h1><a ....></h1>) gebaut wurde.

5. Erfahrungen im Projekt

Im SAP Web Channel Experience Management wurde bereits in der frühen Designphase entschieden, dass man die Barrierefreiheit über eine Umsetzung des ARIA Standards erreichen will, auch wenn dieser zu diesem Zeitpunkt noch kein endgültig verabschiedeter W3C Standard war. Der Hauptgrund für diese Entscheidung war, dass man die Barrierefreiheit der Anwendung relativ einfach (durch die Ergänzung des bestehenden Markups um Rollen und Attribute) erreichen kann. Dies war für dieses Projekt insbesondere wichtig, da hier eine Vielzahl von nicht im HTML Standard enthaltenen Controls benötigt wurde (z. B. Slider, Spin-Button oder Live Regions für Nachrichten).

Allerdings war diese Entscheidung auch mit einem Risiko behaftet. Die Barrierefreiheit der Implementierung über den ARIA Standard ist abhängig von dem jeweiligen Umsetzungsstatus der Betriebssysteme, Browser und der assistiven Technologien. Barrierefreiheit kann nur dann über den ARIA Standard erreicht werden, wenn die entsprechenden Systemumgebungen die Basis dafür bereitstellen. Dies führte dazu, dass die ersten Barrierefreiheitstests der Anwendung ein eher schlechtes Ergebnis aufzeigten. Diese internen Tests einiger Szenarien mit einem Screen Reader wurden Anfang 2010 durchgeführt. Die schlechten Ergebnisse des Tests führten sogar zu Überlegungen, die ARIA Entwicklung rückgängig zu machen, da die technischen Voraussetzungen noch nicht gegeben schienen. Ein erneuter Test Anfang 2011, in der zweiten Phase des Projekts, zeigte dann allerdings, dass die ARIA Unterstützung von Screen Readern und Browsern mittlerweile so gut war, dass die Ziele bzgl. der Barrierefreiheit der Anwendungen erreicht werden konnten.

Zusammenfassend lässt sich feststellen, dass die Anwendung des ARIA Standards im Projekt keine größeren Schwierigkeiten verursachte. Es war mit sehr wenig Aufwand verbunden, die im Rahmen des Projekts entwickelten Controls mit den korrekten ARIA Rollen zu versehen. Auch die Nachrüstung einiger Controls, die zum Zeitpunkt der Entscheidung für den Einsatz des ARIA Standards, schon fertiggestellt waren, erforderte keine tiefergehenden Änderungen an der bestehenden Implementierung .

Für die Anforderungen dieses Projekts war der im ARIA Standard definierte Umfang von Rollen völlig ausreichend. Ein Test der Benutzeroberflächen zeigte, dass der im Moment schon vorhandene Support des ARIA Standards durch die Hersteller von Browsern und Screen Readern schon sehr gut ist. Als Screen Reader wurden in diesem Test JAWS (siehe www.freedomsci.de/serv01.htm, letzter Zugriff 26.5.2011) und NVDA (http://www.nvda.project.org/, letzter Zugriff 26.5.2011) im Zusammenspiel mit dem Internet Explorer und Firefox verwendet. Verglichen mit Versuchen Barrierefreiheit ohne den ARIA Standard über Alternativtexte oder alternative Oberflächen herzustellen, war das Resultat sehr zufriedenstellend.

Für behinderte Nutzer bringt ein Einsatz des ARIA Standards erhebliche Verbesserungen der Bedienbarkeit mit sich. Statt vieler proprietärer Einzellösungen, an die sich der Nutzer wieder auf jeder Web-Seite oder Web-Anwendung neu gewöhnen muss, sind jetzt standardisierte Lösungen möglich, die sich wie die eigentlichen Controls verhalten. D.h. hier wird die Konsistenz der Benutzerinteraktion über verschiedene Lösungen hinweg deutlich erhöht.

Literatur

1. Craig, J. & Cooper, M. (2011). Accessible Rich Internet Applications (WAI-ARIA) 1.0. Online verfügbar unter: http://www.w3.org/TR/wai-aria/ (letzter Zugriff 26.5.2011).
2. Stephanidis, C. & Salvendy, G. (1999). Towards an information society for all: HCI challenges and R&D recommendations. International Journal of Human-Computer Interaction, 11(1), 1-28.
3. Bühler, C. & Stephanidis, C. (2004). European Co-operation Activities Promoting Design for All in information Society Technologies. In: Miesenberger, K., Klaus, J., Zagler, W. & Burger, D. (Hrsg.), ICCHP – Computer Helping People with Special Needs. Volume 3118, 80-87.
4. Jani, R. & Schrepp, M. (2005). Are the Web Accessibility Guidelines applicable and sufficient for Web Applications. In Pruski, A. & Knops, H. (Hrsg.): Assistive Technology: From Virtuality to Reality. IOS Press, 499 – 503.
5. Hardt, A. & Schrepp, M. (2007). Accessibility of Business Web Applications. In: Eizmendi, G., Azkoita, J.M. & Craddock, G.M. (Hrsg.), Challenges for Assistive Technology, 922-927.