

Dokumentation der XML-Spezifikation

Arno & Yasin

30.03.2015

Inhaltsverzeichnis

1 Story	3
2 Dependency	3
3 Anmerkung: Storypoint	5
4 Assets - Video	5
5 Interaction	6
5.1 Quiz	6
5.2 WayChooser	7
5.3 Item	8
5.4 Bitte beachten	8
6 Entscheidungen	9
7 Version	9

1 Story

Das Story-Tag in der XML ist ein MUSS. Dadrin stehen unter anderem:

1. Titel
2. Beschreibung
3. Größe
4. Autor
5. CreationDate
6. Location
7. Radius
8. Storypunkte

Hier werden die einzelnen Storypunkte abgespeichert. Außerdem wird mit abgespeichert, ob man an diesem speziellen Storypunkt Internet hat und ob dieser Punkt überhaupt erreichbar ist. Das kann wichtig sein, sollte ein Anker gewählt werden, der irgendwo über dem Boden liegt und man von einer Empore oder einer anderen Position diesen Anker auslösen muss.

Die Tags “Autor “, “CreationDate “, “Location “und “Radius “werden auf dem Endgerät als Metadaten angezeigt.

“CreationDate “wird im Format : yyyy-mm-dd angegeben.

Dabei ist speziell “Location “der GPS-Punkt, in welcher Stadt die Story spielt (bzw. anfängt). “Radius “ist der Umkreis (angegeben in KM) um den die Location, in der die Story spielt.

2 Dependency

Das alte Konzept der Dependency wurde überarbeitet. Es wurde überarbeitet in:

```

<Storypoint id="X">
  <Container>
    <Punktliste>
      <PunkteRef .../>
    </Punktliste>
    <Itemliste>
      <ItemRef />
    </Itemliste>
  </Container>
  <Container>
    <Punktliste>
      <PunkteRef .../>
    </Punktliste>
    <Itemliste>
      <ItemRef />
    </Itemliste>
  </Container>
  .
  .
  .
  <EndOfStory / >
</Storypoint>

```

Hier unterscheiden wir, welche WEGE wichtig (essenziell) sind, um zu dem Punkt (Storypoint X) zu kommen. Dabei wird in jedem Container gespeichert, welche Storypoints man bis dahin haben muss und welche Items man haben muss. Es kann x-beliebige Container geben.

EndOfStory bedeutet, dass dieser Storypoint ein möglicher Endpunkt der Story ist.

Dies ermöglicht nämlich, dass man Punkte NICHT zusätzlich ausschließen muss, welches unnötigen Aufwand und Verwirrung für den Ersteller beinhaltet.

Mit dem alten Verfahren konnte man nur B,C als Dependency von D wählen, konnte aber nur einen Weg laufen OHNE Berücksichtigung der Items, welche man auf den unterschiedlichen Wegen finden konnte. Es gab

keine Alternativrouten. Es gab lediglich die Möglichkeit anzugeben von welchen anderen Punkten der Punkt X abhängt.

Beispielsweise: X hängt ab von (A,B,C). Allerdings wurde nicht berücksichtigt, dass X evtl. NUR von A abhängen kann. Es gab also die „Öder-Verknüpfung“ nicht. Deshalb wurde das obrige System eingefügt. Es wird nun also in 2 Containern eingetragen, dass X von A,B,C abhängig ist und im zweiten Container lediglich A.

Vorteil: Man erkennt Abhängigkeiten und Itemabhängigkeiten in einem Weg OHNE das ein anderer berücksichtigt wird. Nachteil: XML wird sehr sehr aufgebläht.

3 Anmerkung: Storypoint

Wie man sieht werden die Storypoints doppelt abgespeichert:

Einmal mit der Erreichbarkeit und ob es dort Internet gibt.
Ein andermal wie die einzelnen Punkte zusammenspielen.

Der Grund ist, dass der jeweilige Player gesondert feststellen kann ob er vorher wichtigen Content runterladen muss (Tag: Story) und das der Server gesondert testen kann, ob die Story überhaupt spielbar ist (Tag: Dependency).

4 Assets - Video

Es gibt verschiedenste Assets in ARML2.0, die alle ohne weiteres genützen werden können. Das Asset "Video" haben wir mit eingefügt:

```
<xsd:complexType name="VideoType">
  <xsd:complexContent>
    <xsd:extension base="VisualAsset2DType">
      <xsd:sequence .../>
      <xsd:element name="Href">
        <xsd:complexType>
```

```

        <xsd:attribute ref="xlink:href" / />
        .
        .
        .
</Storypoint>

```

sowie

```

<xsd:element name="Video" type="VideoType" substitutionGroup="VisualAsset2D" />

```

Video ist also momentan ein VisualAsset2D.

Wir haben es dann unter den "ScreenAnchorType" gelegt, genauer "AnchorType":

```

<xsd:complexType name="ScreenAnchorType">
  <xsd:extension base="AnchorType" />
  .
  <xsd:element ref="Video" maxOccurs="1" minOccurs="0" />

```

Ein Video kann maximal einmal vorkommen in einem Asset.

5 Interaction

Um mit dem Gerät zu interagieren und Entscheidungen zu treffen haben wir Interaktionen des ARML2.0-Standarts hinzugefügt. Hierbei haben wir als Überbegriff "Interaction" gewählt. Weitere Interaktionen werden in diese XSD mit aufgenommen. Bisher haben wir folgende Interaktionen definiert:

1. Quiz
2. WayChooser
3. Item

5.1 Quiz

Ein Quiz ist beispielsweise so aufgebaut:

```

<Quiz id="Quiz_ 1">
  <FeatureRef xlink:href="# Punkt_ A_ Feature"/>
  <OnTrue xlink:href="# Feature_ A_ 1"/>
  <OnFalse xlink:href="# Feature_ A_ 2"/>
  <Question >
    Frage
  </Question >
  <Answer id="Ä1">
    <Text .../>
    <status .../>
  </Answer >
  .
  .
  .

```

Es kann nur eine Frage geben, aber x-beliebige Antworten.

Die Tags: `OnTrue` und `OnFalse` sind Optional und verweisen auf eine andere Feature, falls man eine Entscheidung zwischen 2 Sachen hat.

Sollte man sich zwischen mehr als 2 Sachen entscheiden müssen gibt es den `WayChooser`:

5.2 WayChooser

```

<WayChooser id="WayChooser_ 1">
  <FeatureRef xlink:href="# Punkt_ A_ Feature"/>
  <Question >
    Frage
  </Question >
  <Answer id="Ä2">
    <Text .../>
    <status .../>
    <ItemRef xlink:href="# Item_ 1" / >
  </Answer >
  <Answer id="Ä3">

```

```

    <Text .../>
  <status .../>
    <ItemRef xlink:href="# Item_ 2" / >
    <FeatureRef xlink:href="# Punkt_ B_ E2" / >
</Answer >
.
.
.

```

Hierbei kann man also nach einer Entscheidung ein Item aufgenommen werden oder gar auf eine andere Interaktion verwiesen werden.

Es muss hier nicht explizit auf das nächste Feature verwiesen werden, da die Abhängigkeiten in dem Tag "Dependency" geregelt sind, d.h. man muss lediglich das Item aufnehmen und die Interaktion verlassen. Sollte nach dieser Interaktion das bisherige Feature weitergehen (zum Beispiel ein erklärendes Video was man mit dem Item machen sollte), muss dieses Feature gesondert deklariert werden und das Item darauf verweisen mittels FeatureRef.

5.3 Item

Ein Item ist relativ übersichtlich aufgebaut:

```

<Item id="Item_ 1">
  <Description >
    Beschreibung
  </Description >
  <FeatureRef xlink:href="# Punkt_ C_ E1" / >
  <IsCollected >
    True oder False
  </IsCollected >
</Item>

```


5.4 Bitte beachten

Bei allen gilt: FeatureRef unter einem Interaction-Type verweist auf ein Feature, nachdem diese Interaktion abgespielt werden soll..

FeatureRef in einem Element eines Interaction-Types verweist auf das Feature, welches danach kommen soll.

Desweiteren MUSS jede "id" eindeutig sein.

6 Entscheidungen

Sollte während eines Assets (eines Videos) eine Entscheidung auftreten, so muss das laufende Feature ab dem Zeitpunkt der Entscheidung enden und auf weiterführende Features mit der restlichen Logik des vorangegangenen Features verweisen. So wird ein Video, welches eine Entscheidung beinhaltet in (Anzahl der Entscheidungen:) x Features aufgeteilt:

1. Feature 1: Bis zur Entscheidung
2. Feature 1_ 1: Nach der Entscheidungsoption 1
3. Feature 1_ 2: Nach der Entscheidungsoption 2
4. ...

7 Version

Damit die Versionen der XSD mit der der XML übereinstimmen gibt es im Header immer ein Tag mit dem Namen: "annotation ". In diesem Tag gibt es das Tag "documentation", welches u.a. das Attribut: "Revision" hat. Dies zeigt die aktuelle Version an. Die Version der XSD sollte immer aktuell sein.