EasyChair Preprint

№ 6224

Elements of Document Type Definition in Maiar Game

Frank Appiah

August 4, 2021

ELEMENTS OF DOCUMENT TYPE DEFINITION IN MAIAR GAME .

FRANK APPIAH.

KING'S COLLEGE LONDON, SCHOOL OF ENGINEERING, LONDON, ENGLAND, UNITED KINGDOM.

appiahnsiahfrank@gmail.com. | frank.appiah@kcl.ac.uk

Extended Abstract[+].This is a thesis on declaration of definitions of a document that models the world representation of a maiar game.A Maiar[1, 2] DTD(document type definition) declares definitions that are revelant for the document content of the game. This is means to validate the game content of document instance against DTD. A standalone document without a valid DTD can yield different game situations.

Keywords. document, type, definition, validation, content, game, elements, schema.

Year of Study: 2017 **Year of Publication: 2020**

1 ***AFFILIATE. UNIVERSITY OF LONDON, KING'S COLLEGE LONDON, DEPARTMENT OF INFORMATICS, LONDON, UK.**

1 INTRODUCTION

In modern game design, documenting game plays as an external entity is essential to the development as well. The whole idea is to create a game factory of states and actions in a world.

A schema[3] defines a document type or class of documents by imposing a set of constraints on document instances. This thesis postulates that document of type, *map* introduces the whole content of the game world.

In order of the world to be used during runtime it needed to be converted into an Object. The chosen object type for the parsing was a graph of rooms for the room representation and a map of commands for the global commands. For parsing the XML[3], the use of the Jdom library has proven to be the best choice because of the tree form of the XML file it offers, and the access methods to the XML file. First the XML is read using a SAX builder, which parses the XML to a tree. Then each room is parsed on its own using the getChildren command which returns a list of the chosen element
type.

2 WORLD DOCUMENT ELEMENTS

The world element of game is based on element tag types and content data attributes in DTD schema[3]. The *map* element has about 4 content data variables with a name attribute. Each context of element tag can have different elements as sub-types. The DTD schema is made up of element tags (sub-type elements) and content datatypes of CDDATA.

This section will discuss a postulates of about 21 document elements of Maiar game. This includes map element, exit element, rooms element, intro element, items element, look element, action element, directions element, requirement element, commands element, global element, satisfied element, nonSatisfied element, north element, south element, east element, west element and effect element.

This DTD structure of a game world as represented as a Document Type Definition schema is shown below:

2.1 Type Map Definition

The three basic definition of Type map element are:

- Atrribute List[1]: `name`
- **Attribute Tag:** `<!ATTLIST map name CDATA #IMPLIED>`
- Element Types : `exit, global, rooms, entry`.
- **Tag Element**: It is denoted as a map tag:
  `<!ELEMENT map(exit | global| rooms| entry)*>`

2.2 Type Rooms Definition

The basic definition of Type rooms element are:

- Element Type: `room`
- **Tag Element**: A rooms tag is denoted as:
  `<!ELEMENT rooms (room)*>`

2.3 Type Room Definition

The basic definition of Type room definition are:

- Atrribute List: `id, name`

[1] ATTRIBUTE LIST: ATTLIST rules are always terminal because XML attributes cannot have a complex structure.

- **Attribute Tag:**<!ATTLIST room
  id CDATA[2] #IMPLIED
  name CDATA #IMPLIED>
- **Element Types:** commands, directions, items, look, intro
- **Tag Element**: A room tag is denoted as:
  <!ELEMENT room(commands | directions | items| look| intro)*>

2.4 Type Intro Definition

The basic definition of Type intro element are:

- Attribute List: message
- **Attribute Tag:** <!ATTLIST message CDATA #IMPLIED>
- **Tag Element**: A intro tag element is denoted as:
  <!ELEMENT intro EMPTY>

2.5 Type Look Definition

The basic definition of Type Look element are:

- Attribute Type: description
- **Attribute Tag:** <!ATTLIST description CDATA #IMPLIED>
- **Tag Element**: A Look tag is denoted as:
  <!ELEMENT Look EMPTY>

2.6 Type Items Definition

The basic definition of Type items element are:

- Element Type: item
- **Tag Element**: A items tag is denoted as:
  <!ELEMENT items(item)*>

2.7 Type Item Definition

The basic definition of Type item element are:

- Attribute List: name

[2] CDATA: Character Data

- **Attribute Tag:** `<!ATTLIST name CDATA #IMPLIED[3]>`
- **Tag Element**: A item tag is denoted as:
  `<!ELEMENT item EMPTY>`

2.8 Type Directions Definition

The basic definition of Type directions element are:

- Element Types: `north, south, east, west`
- **Tag Element**: A directions tag is denoted as:
  `<!ELEMENT directions( north|south|east|west)>`

2.9 Type North Definition

The basic definition of Type north element are:

- Attribute List: `roomId`
- **Attribute Tag:** `<!ATTLIST roomId CDATA #IMPLIED>`
- **Tag Element**: A north tag is denoted as:
  `<!ELEMENT north EMPTY>`

2.10 Type South Definition

The basic definition of Type south element are:

- Attribute List: `roomId`
- **Attribute Tag:** `<!ATTLIST roomId CDATA #IMPLIED>`
- **Tag Element**: A south tag is denoted as:
  `<!ELEMENT south EMPTY>`

2.11 Type East Definition

The basic definition of Type east definition are:

- Attribute List: `roomId`
- **Attribute Tag:** `<!ATTLIST roomId CDATA #IMPLIED>`
- **Tag Element**: A east tag is denoted as:
  `<!ELEMENT east EMPTY>`

[3] IMPLIES: Attribute is neither required nor does it have a default value.

2.12 Type West Definition

The basic definition of Type west element are:

- Attribute List: `roomId`
- **Attribute Tag:** `<!ATTLIST roomId CDATA #IMPLIED>`
- **Tag Element**: A west tag is denoted as:
  `<!ELEMENT west EMPTY>`

2.13 Type Commands Definition

The basic definition of Type commands element are:

- Element Type: `command`
- **Tag Element**: A commands tag is denoted as:
  `<!ELEMENT commands (command)*>`

2.14 Type Command Definition

The basic definition of Type command element are:

- Attribute List: `name`
- **Attribute Tag:** `<!ATTLIST name CDATA #IMPLIED>`
- **Element Type**: action
- **Tag Element**: A command tag is denoted as:
  `<!ELEMENT command (action)*>`

2.15 Type Action Definition

The basic definition of Type action element are:

- Element List: `requirement, effect`
- **Tag Element**: A action tag is denoted as:
  `<!ELEMENT action (requirement, effect)*>`

2.16 Type Effect Definition

The basic definition of Type effect Definition are:

- Attribute List: `operator, value, parameter`
- **Attribute Tag:** `<!ATTLIST`
  `operator CDATA #IMPLIED`
  `value CDATA #IMPLIED`
  `parameter CDATA #IMPLIED>`

- **Tag Element**: A effect tag is denoted as:
  `<!ELEMENT effect EMPTY>`

2.17 Type Requirement Definition

The basic element of Type requirement Definition are:

- Attribute List: `value, parameter`
- **Attribute Tag:** `<!ATTLIST value CDATA #IMPLIED parameter CDATA #IMPLIED>`
- Element Types: `notSatisfied, satisfied,`
- **Tag Element**: A requirement tag is denoted as:
  `<!ELEMENT requirement (notSatisfied | satisfied)*>`

2.18 Type Notsatisfied Definition

The basic definition of Type Notsatisfied Definition are:

- Element Type: `action`
- **Tag Element**: A Notsatisfied tag is denoted as:
  `<!ELEMENT Notsatisfied (action)*>`

2.18 Type satisfied Definition

The basic definition of Type satisfied definition are:

- Element Type: `action`
- **Tag Element**: A satisfied tag is denoted as:
  `<!ELEMENT satisfied (action)*>`

2.19 Type Global Definition

The basic definition of Type global element are:

- Element Type: `command`
- **Tag Element**: A global tag is denoted as:
  `<!ELEMENT global (command)*>`

2.20 Type Exit Definition

The basic definition of Type exit element are:

- Attribute Types: `id, room`
- **Attribute Tag:** `<!ATTLIST value CDATA #IMPLIED parameter CDATA #IMPLIED>`
- **Tag Element**: A exit tag is denoted as: `<!ELEMENT exit EMPTY>`

4 CONCLUSION

This work concludes on document analysis on Maiar DTD elements as described above. This postulates are used in the validating of game scenario in order not to yield untold game situations. XML technology is proven to be perfect in describing states and actions in a game environment like Maiar.

Compliance with Ethical Standards:

(In case of funding) Funding: This is research is funded by King's Alumni Group, Association of Engineering/postgraduate grant with ISAreference grant number: 204424 20821845.

Conflict of Interest:

Author, Dr. Frank Appiah declares that he has no conflict of interest .

REFERENCES

1. Appiah F. (2009/10), Electronic Game Design with XML Technology, KCL Msc Group Project, London, England.
2. Appiah F. (2011/12), Design and Implementation of a Jxta-based Game Software in Java.
3. Berthold D.(2003), Modeling Business Objects with XML Schema. Elsevier Science USA, Morgan Kaufman Publication.