NLQTI: Dutch Profile for Question and Test Interoperability

Tests

Version: V1.0 (November 2012)

Table of Contents

Colophon

Project team:	Jim Bijlstra; Jeroen Hamers; Marjolijn van Hooff; Erik Siegel
Author(s):	Erik Siegel
Consulted experts:	Wouter Huijnink; Bieke vd Korst
Consulted organizations:	Andriessen; Boom Test Uitgevers; Bureau ICE; Citaverde College; CITO; College van Examens; De Rode Planeet; Deviant; Edu'Actief; Efinity; Malmberg; Noordhoff Uitgevers; Orange11; Paragin; Platform VVVO;Roadside; SURF; ThiemeMeulenhoff; Threeships;

Document history

Versie	Datum	Omschrijving
V0.91	April 2011	Eerste uitwerking voor plateau 2
V0.92	Mei 2012	Kleine tekstuele bijstellingen n.a.v. expert groep bijeenkomst
V0.93	Sep 2012	Translated to English
V1.0	November 2012	Small textual changes and typos. Abbreviation changed to NLQTI. Colophon added. Final version.

1 Inleiding

This document is part of the technical documentation set for the Dutch Profile for Question and Test Interoperability, a.k.a. NLQTI. The full set of documents is listed in [NLQTI-ICS] (see sect. 1.1/pg. 3). Their goal is to supply content and software developers with enough information to implement this profile.
A common introduction and general information can be found in the document *"NLQTI: Dutch Profile for Question and Test Interoperability – Introduction and Common Sections"*.

This part of the profile describes:

- QTI Tests. Technically these are the QTI documents with root element `<assessmentTest>`
- The response and feedback processing for tests

1.1 Additional documents

1.1.1 Profile documents

The Dutch Profile for Question and Test Interoperability consists of the following documents:

[NLQTI-AB]	NLQTI: Dutch Profile for Question and Test Interoperability - Algemene beschrijving toepassingsprofiel op basis van IMS QTI v2.1 *Functional description of the profile,* ***Dutch only****.*
[NLQTI-ICS]	NLQTI: Dutch Profile for Question and Test Interoperability – Introduction and common sections *Common parts of the profile.*
[NLQTI-ITEM]	NLQTI: Dutch Profile for Question and Test Interoperability - Items *Description of items within this profile.*
[NLQTI-TEST]	Dutch Profile for Question and Test Interoperability - Tests *Description of tests within this profile*
[NLQTI-CP]	NLQTI: Dutch Profile for Question and Test Interoperability - Content Packaging *Describes the way items and tests should be packaged according to this profile.*

1.1.2 Other documents and sources of information

A list with further additional documents and other sources of information can be found in [NLQTI-ICS].

2 QTI Tests functional description

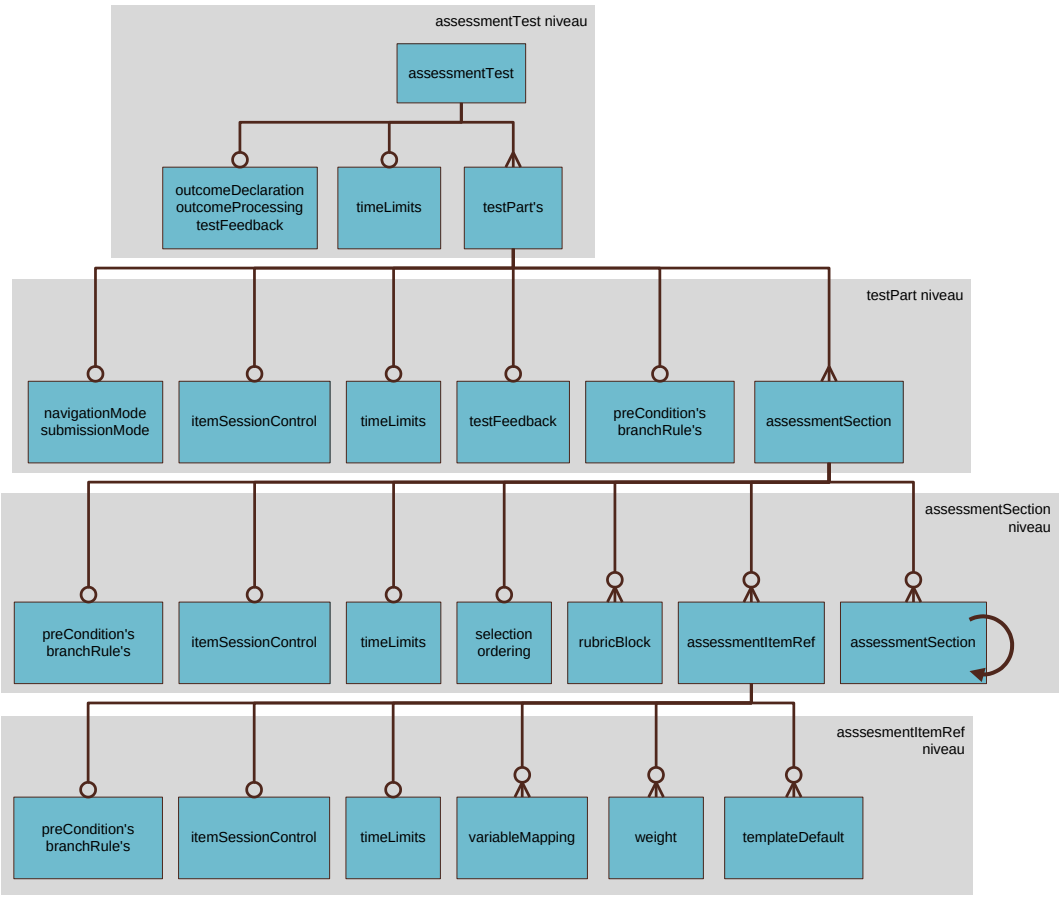
A QTI test can become quite complicated. You can influence things like:

- How a test can be divided into sections. Sections consist of (references to) items and/or further sub-sections.
- Every referenced item and (sub)section has many properties. For instance the time the learner is allowed to spend, how often it can be done, when and how feedback is shown, etc.
- Before it is presented to the learner, a test (or section of a test) can compose itself. For instance from a section containing 100 items, 10 are selected at random and presented to the learner
- Adaptive questioning: A test can specify a dynamic path through the content, based on the performance of the learner. For instance, if you have more than 10 points at this stage of the test, skip the next section and go directly to the bonus section.
- Perform computations during and at the end of a test, for instance to calculate an overall score.

This profile reduces the complexity and eases implementation of software handling QTI. To underpin the choices made, this chapter handles QTI tests in more detail and presents the choices made on an overview level.

2.1 Anatomy of a QTI Test

If we look at a QTI Test in some more detail, this is how it is structured:



Top to bottom:

- **`assessmentTest` level:** Coordinating level for the full test.
  - `outcomeDeclaration`, `outcomeProcessing`, `testFeedback`: Handles results processing and feedback for the full test. The options here are however considerable less functional than those for items.
    For instance, the "template mechanism" we used in items is lacking. As a consequence, we cannot use fixed processing templates and have to prescribe its content.

- `timeLimits`: Time limits for the test as a whole
- `testPart`: A test consists of one or more `testPart`'s:

- **`testPart` level:** Main section level for a test.
  - `navigationMode`, `submissionMode`: Properties for determining how a learner can navigate through the section (linear or at random) and when results processing is performed (only at the end or after every item)
  - `itemSessionControl`: Properties for the underlying `assessmentItem's`. For instance, how often can a learner retry an item.
  - `timeLimits`: Time limits for this `testPart`
  - `testFeedback`: Specific feedback for this `testPart`
  - `preCondition`, `branchRule`: Properties to make this `testPart` adaptive. Do we have to show this `testPart` at all and/or what's the next `testPart` to show.
  - `asssessmentSection`: A `testPart` consists of one or more `assessmentSection's`
- **`assessmentSection` level:** Sub-section level for a test. An `assessmentSection` can contain `assessmentSection's` recursively.
  - `preCondition`, `branchRule`: Properties to make this `assessmentSection` adaptive. Do we have to show this `assessmentSection` at all and/or what's the next section to show.
  - `itemSessionControl`: Properties for the underlying `assessmentItem's`. It overrules settings made on higher levels.
  - `timeLimits`: Time limits for this `assessmentSection`
  - `selection`, `ordering`: Properties for ordering and selecting parts of this section before the test is done. For instance, do we need a random subset of this section and should its order be randomized.
  - `rubricBlock`: An optional block with additional text or other information belonging to this `assessmentSection`. For instance the main text all items are about.
  - `assessmentItemRef`: Zero or more references to items
  - `assessmentSection`: Zero or more `assessmentSection's`.
- **`assessmentItemRef` level:** A reference to an item with several test specific properties:
- A reference to an item with the following settings:
  - `preCondition`, `branchRule`: Properties to make this item adaptive. Do we have to show this at all and/or what's the next item/section to show.
  - `itemSessionControl`: Properties for this item. It overrules settings made on higher levels.
  - `timeLimits`: Time limits for his specific item
  - `variableMapping`: An addition to map the item's outcome variables to other names (to aliases).
  - `weight`: Property to give the score for this item a certain weight in the overall score..
  - `templateDefault`: Property for the template facility of items. Templates are not allowed in this profile.

2.2 Functional description

The following choices were made to limit the complexity of tests for this profile:

1) Limit all the options QTI offers for structuring a test to: A single `testPart` ➔ a single (main)`assessmentSection` ➔ zero or more (sub)`assessmentSection's` (prohibiting further nesting of `assessmentSection's`).
   Underpinning:
   - A QTI test's structure can become arbitrarily complex. This will probably not be supported by all authoring and rendering systems. Limiting the structure's complexity greatly enhances the chances for support. The structure proposed is used in the Netherlands.
   - A first layer of `assessmentSection` is necessary because a `rubricBlock` can be defined at `assessmentSection` level only. This `rubricBlock` can contain information for the learner for the complete test (if a `rubricBlock` was defined at the `testPart` level also, this layer wouldn't have been necessary).
   - The second `assessmentSection` layer is necessary to keep questions together in a group. This creates the option for arranging singular questions and question groups (for which the questions must stay together).
2) Limit the options for setting time limits to the full test and the item level. Do not allow setting time limits on the level(s) in between.
   Underpinning: Setting limits on all levels is not an often requested requirement. When allowed

it will increase complexity handling timers and time limits. Time limits at top and bottom level seems enough for now.

3) Limit the options for feedback to the highest level (`assessmentTest`) only. Only allow feedback at the end of the test based on passed/failed.
Underpinning: The requirement for more complex feedback during a test absolutely exists. However, allowing this would also mean a much more complex results processing. If we limit it as proposed, we can work with fixed results processing. If we don't, all engines must support the QTI programming language, something we're trying to avoid to ease implementation.

4) Do not allow the options for adaptive testing
Underpinning: Allowing adaptive tests increases the complexity of the authoring and rendering systems manifold. This conflicts with the points of departure for this profile (see [NLQTI-ICS]).

5) Only allow the following results processing:
   - Computing the score based on the weighted total of the individual item scores
   - Controlling feedback

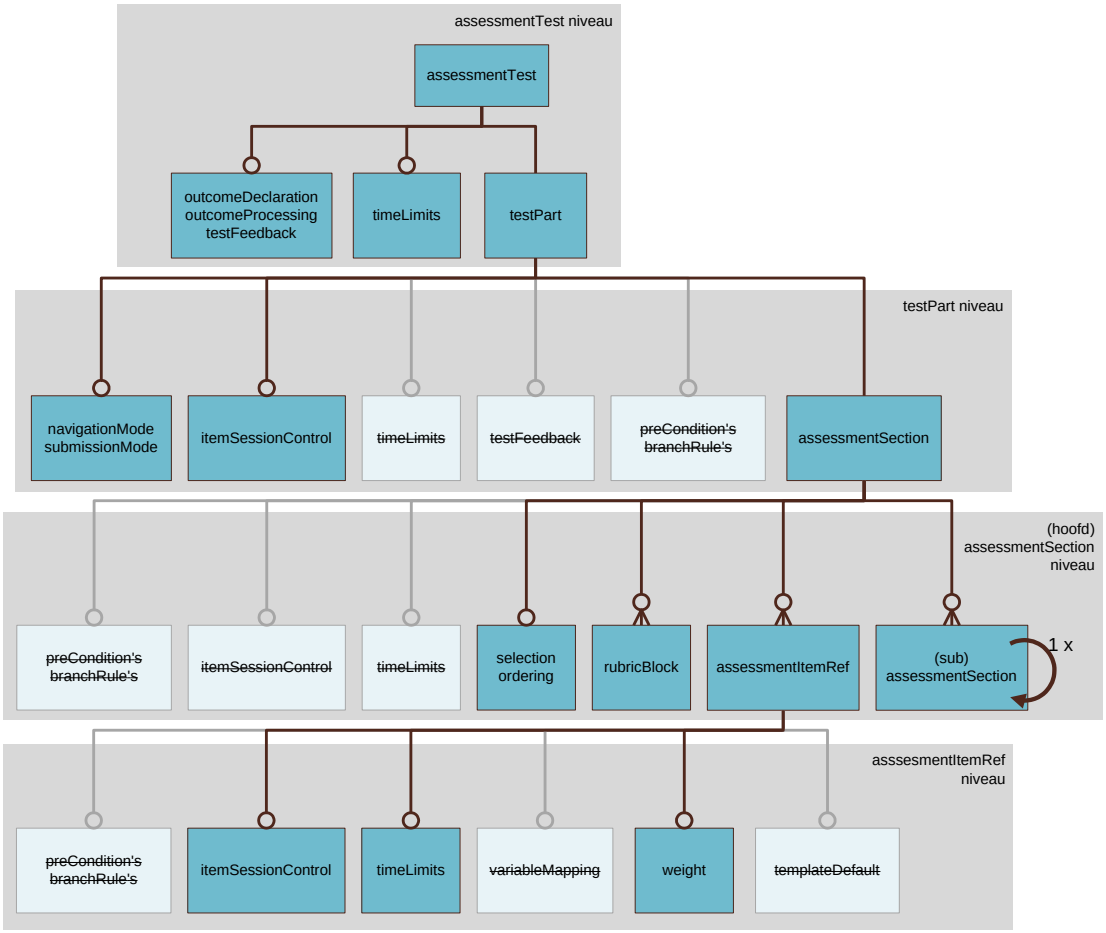
   Underpinning: There seems to be no requirement for more complex computations of the end score. Allowing simple feedback is a user requirement.

6) Limit the options for specifying the `itemSessionControl` to the main `testPart` (highest level, for all items) and the individual item references.
Underpinning: Because we limited the test's structure (see point 1 above), detailed control is unnecessary and will only result in superfluous complexity. Setting this only once with the option to override it on an item level suffices.

7) A reference to a question (`assessmentItemRef`) is only allowed to contain a single scoring weight (for computing the overall score).
underpinning: We only have a single overall score, so defining more than one weight is unnecessary.

2.3 Anatomy of a test according to the profile

Combining the anatomy of a test with the functional descriptions, a test within this profile looks as follows (left out parts are show in a lighter color):



Examples of tests according to the profile can be found in the accompanying example files..

3 QTI test structure

3.1 assessmentTest level

A QTI test is a separate XML document for which the root element is always `<assessmentTest>`.

Attributes for `<assessmentTest>`:

Naam	Prf?	M	Type	Remarks
identifier	Yes	1	string	Preferably, an identifier should adhere to the rules stated in [KN-PID]
title	Yes	1	string	
toolName	Yes	?	string256	See [NLQTI-ICS], sect. 4.3
toolVersion	Yes	?	string256	

Sub-elements of `<assessmentTest>`:

Naam	Prf?	M	Remarks
outcomeDeclaration	Lim	*	See sect. 5.2.1/pg. 12
timeLimits	Yes	?	See sect. 6.1/pg. 17
testPart	Lim	1	See sect. par. 3.2/pg. 8
outcomeProcessing	Lim	?	See sect. 5.2.2/pg. 14
testFeedback	Lim	*	See sect. 5.2.3/pg. 16

3.2 testPart level

The first level for QTI tests is `testPart`. The profile only allows a single `testPart`.

Attributes of `<testPart>`:

Naam	Prf?	M	Type	Remarks
identifier	Yes	1	string	
navigationMode	Yes	1	string	Values: – `linear`: The learner is allowed to handle the items ordered only. He/she is not allowed to navigate between items at random. – `nonlinear`: The learner is allowed to navigate between items at random
submissionMode	Yes	1	string	Values: – `individual`: The results of an item are handled when finishing the item – `simultaneous`: The results of all items are handled all together at the end of the test Watch out: `simultaneous` mode also means that feedback on item level will not occur! All results processing, including feedback handling, is postponed until the end of the test.

Sub-elements of `<testPart>`:

Naam	Prf?	M	Remarks
preCondition	No		
branchRule	No		
itemSessionControl	Yes	?	See sect. 6.2/pg. 17
timeLimits	No		
assessmentSection	Lim	1	See sect. 3.3/pg. 8
testFeedback	No		

3.3 Main assessmentSection level

The profile defines underneath the `testPart` level a single `<assessmentSection>` element. This is called the main `assessmentSection` level.

Attributes of the main <assessmentSection> element:

Naam	Prf?	M	Type	Remarks
identifier	Yes	1	string	
required	No			No meaning/implication for the main level
fixed	No			No meaning/implication for the main level
title	Yes	1	string	
visible	Lim	1	boolean	Always true for the main level (has no meaning/implication but is a mandatory attribute)
keepTogether	No			No meaning/implication for the main level

Sub-elements of the main <assessmentSection> element:

Naam	Prf?	M	Remarks
preCondition	No		
branchRule	No		
itemSessionControl	No		
timeLimits	No		
selection	Yes	?	See sect. 4.1/pg. 11
ordering	Yes	?	See sect. 4.2/pg. 11
rubricBlock	Yes	*	
assessmentItemRef	Yes	*	See sect. 3.5/pg. 10
assessmentSection	Yes	*	See sect. 3.4/pg. 9

- According to the underlying QTI specification, an assessmentSection can also be empty (no items or sub-sections). The profile disallows this: There has to be at least one assessmentItemRef or assessmentSection element present in the main assessmentSection.

3.4 sub assessmentSection level

Multiple <assessmentSection> elements are allowed underneath the main <assessmentSection> level. This is called the sub <assessmentSection> level.

Attributes of the sub <assessmentSection> element:

Naam	Prf?	M	Type	Remarks
identifier	Yes	1	string	
required	Yes	?	boolean	Meaning: When doing a random pre-selection (sect. 4/pg. 11), this section must always be selected.
fixed	No			Defining a fixed position for some components of a test is not allowed. This is consistent with the choices made for randomizations inside items.
title	Yes	1	string	
visible	Yes	1	boolean	Is the fact that there is a section visible to the learner? - true: The rendering engine is allowed to show the existence of the section to the learner, for instance in some kind of hierarchical overview. - false: The rendering engine is not allowed to show the existence of the section to the learner. All content in this section must be shown as part of the surrounding section.
keepTogether	Yes	?	boolean	- true (default): The content of this section must be kept together. - false (applicable only for visible="false"): The content in this section can be mingled with the content of the surrounding section

Sub-elements of the sub <assessmentSection> element:

Naam	Prf?	M	Remarks
preCondition	No		
branchRule	No		
itemSessionControl	No		
timeLimits	No		
selection	Yes	?	See sect. 4.1/pg. 11
ordering	Yes	?	See sect. 4.2/pg. 11
rubricBlock	Yes	*	
assessmentItemRef	Lim	+	The QTI specification allows empty <assementSection>'s. This profile however excludes empty sections. See sect. 3.5/pg. 10
assessmentSection	No		Deeper levels of <assessmentSection>'s is not allowed.

- The QTI specification allows empty <assessmentSection>'s. This profile disallows this: An <assessmentSection> must always contain at least one <assessmentItemRef>.

3.5 assessmentItemRef level

The assessmentItemRef level contains the link to the actual item. An item is a separate XML document as described in [NLQTI-ITEM].

Attributes of the sub <assessmentItemRef> element:

Naam	Prf?	M	Type	Remarks
identifier	Yes	1	string	
required	Yes	?	boolean	Meaning: When doing a random pre-selection (sect. 4/pg. 11), this item must always be selected.
fixed	No			Defining a fixed position for some components of a test is not allowed. This is consistent with the choices made for randomizations inside items.
href	Yes	1	uri	Reference to the item. References are defined in [NLQTI-ICS], sect. 4.4
category	No			

Sub-elements of the sub <assessmentItemRef> element:

Naam	Prf?	M	Remarks
preCondition	No		
branchRule	No		
itemSessionControl	Yes	?	See sect. 6.2/pg. 17
timeLimits	Yes	?	See sect. 6.1/pg. 17
variableMapping	No		
weight	Lim	1	See sect. 3.5.1/pg. 10
templateDefault	No		

3.5.1 Weight of an item

- To allow computation of a more balanced total score for a test, items can be assigned a "weight". For instance, the score for question A is twice as important as the score for question B. The <assessmentItemRef> element has a <weight> sub element for this.
- This profile allows the assignment of a single weight to an item only. This weight is always called WEIGHT.
- The specification of the <weight> element probably contains a bug: <weight> has a mandatory attribute called value of data type xs:double. However, it's also mandatory to add a value to the <weight> element of type xs:double. Both could be interpreted as the weight of the item…
  To circumvent this ambiguity (and until the QTI specification specifies what to do about this), this profile dictates that both options (attribute and element value) must be used and that both values must be the same. For instance:

```
<weight identifier="WEIGHT" value="2">2</weight>
```

4 Dynamic tests

The QTI specification allows dynamic tests. That is, before the test is shown to the learner, a selection of the content is taken and randomization is applied. The guiding elements for this are `<ordering>` and `<selection>`, both sub-elements of `<assessmenbtSection>`.

4.1 <selection> element

An `<assessmentSection>` is allowed to have an optional `<selection>` element to define how many components of the `<assessmentSection>` will be part of the final test. A component can be an `<assessmentSection>` or `<assessmentItemRef>`.

Attributes of the `<selection>` element:

Naam	Prf?	M	Type	Remarks
`select`	Lim	1	integer	This defines the number of components (`<assessmentSection>` or `<assessmentItemRef>`) in the final test. It must (of course) be less than the number of components available.
`withReplacement`	No			Using an item twice (or more) is not a functional requirement.

- An `<assessmentSection>` or `<assessmentItemRef>` must always be part of the final test if the attribute `required="true"` is set.
- QTI allows additional attributes and elements on a `<selection>` element to cater for more complex selection algorithms. This profile disallows this.

4.2 <ordering> element

An `<assessmentSection>` is allowed to have an optional `<ordering>` element to define the randomization of the content.

Attributes of the `<ordering>` element:

Naam	Prf?	M	Type	Remarks
`shuffle`	Yes	1	boolean	

- QTI allows additional attributes and elements on an `<ordering>` element to cater for more complex selection algorithms. This profile disallows this.

5 Response en feedback processing

This profile limits the response and feedback processing for tests to:

- Computing a weighted total score
- Guiding the feedback (for the test as whole, not for parts)

5.1 Functional description

5.1.1 Score

This profiles defines the score for a test as:

- The score of an item is always a floating point number in between 0.0 (wrong/bad) and 1.0 (correct/very good).
- The score for a test is computed from the weighted scores of the individual items
- Passing the score to the environment is done using an `<outcomeDeclaration>` with the fixed name `SCORE` (sect. 5.2.1.2/pg. 13).

5.1.2 Feedback

This profile limits feedback to the learner to no feedback or feedback based only on whether the answer is correct or incorrect. The feedback can only appear at the end of the test, not during the test. It is not allowed to add feedback based on more complex criteria like for instance the number of times the learner tried to answer or the time passed.
Most important reasons for these limitation:

- This profile limits itself to testing (in contrast to exercising). Feedback plays a much less prominent role in testing then it does in exercising.
- Allowing more complex feedback would have caused a disproportional increase in complexity of the profile (causing more complex implementations of authoring systems, rendering engines, etc.). Allowing QTI tests to change their behavior based on criteria like the number of tries or the time passed would mean allowing complex and non-standard `<responseProcessing>`, something we're trying to avoid in this profile.

So this profile limits feedback for tests to:

- No feedback: There are no feedback elements in the test
- Singular feedback:
  - There is feedback present for "failed"
  - There is optional feedback present for "passed"
- More detailed feedback, based on other criteria or based on the actual answers given, is not allowed.
- Feedback is allowed at the end of the test only.

5.2 Technical implementation

The definition of response and feedback processing is scattered over a number of different sections/elements within an item:

- The `<outcomeDeclaration>` declarations
- The `<outcomeProcessing>` section
- The `<testFeedback>` definitions

*5.2.1 Elements: **<outcomeDeclaration>***

5.2.1.1 General

- An `<outcomeDeclaration>` is an internal variable in a QTI test. For instance:

```
<outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float"
   normalMinimum="0.0" normalMaximum=1.0"/>
```

- Different `<outcomeDeclaration>'s` than defined in the profile are not allowed.

- Attributes of <outcomeDeclaration>:

Name	Prf?	M	Type	Remarks
identifier	Lim	1	identifier	Three possible values: "SCORE", "FEEDBACK", "FEEDBACK_THRESHOLD"
cardinality	Lim	1	cardinality	Always "single"
baseType	Lim	1	type	For "SCORE", "FEEDBACK_THRESHOLD": "float" For "FEEDBACK": "identifier"
view	Lim	?	view	Don't use or use the fixed value "candidate"
interpretation	Yes	?	string	
longInterpretation	No			
normalMaximum	Lim	?	float	For SCORE always "1.0" Others: Don't use
normalMinimum	Lim	?	float	For SCORE always "0.0" Others: Don't use
masteryValue	No			

- Child elements of <outcomeDeclaration>:

Name	Prf?	M	Remarks
defaultValue	Lim	?	Only when specified below
matchTable	No		
interpolationTable	No		

5.2.1.2 Score

- A test must always define an <outcomeDeclaration> called SCORE of type float.
- The attributes normalMimnimum and normalMaximum must always be present (necessary for the correct processing of scores on optional higher levels)
- So the declaration will always be:

```
<outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float"
   normalMinimum="0.0" normalMaximum=1.0"/>
```

- Because it's a numeric variable, its default value is zero. In contrast to what happens in some of the QTI examples, you do not have to set this explicitly.
- The value of the SCORE <outcomeDeclaration> is always in between 0.0 (wrong/incorrect) and 1.0 (very good/correct).
- A test for which all items have no score (for instance a test composed of all <extendedTextInteraction> items, always yields the score 1.0
- A test for which all weights are 0.0, always yields the score 1.0.
- It's the task of the response processing to give the SCORE <outcomeDeclaration> the right value.

5.2.1.3 Feedback

- In case of feedback, there must be an <outcomeDeclaration> with the fixed name FEEDBACK of type identifier:

```
<outcomeDeclaration identifier="FEEDBACK" cardinality="single" baseType="identifier"/>
```

- The FEEDBACK <outcomeDeclaration> must take one of the following three values only:

Value	Meaning
(null)	No answer given (yet)
RESULT_OK	Test passed
RESULT_NOTOK	Test failed

- Determining whether a test is passed or failed is done using the FEEDBACK_THRESHOLD <outcomeDeclaration> (see sect. ☐/pg. 13)
- It's the task of the response processing to give the FEEDBACK <outcomeDeclaration> the right value.

5.2.1.4 Feedback threshold

- In case of feedback, the response processing must, to display the right feedback, know when to consider the test passed or failed. This is done by defining a special `<outcomeDeclaration>` with the name `FEEDBACK_THRESHOLD`:

```
<outcomeDeclaration identifier="FEEDBACK_THRESHOLD" cardinality="single"
                                                                  baseType="float">
  <defaultValue>
    <value>0.75</value>
  </defaultValue>
</outcomeDeclaration>
```

- The `FEEDBACK_THRESHOLD` `<outcomeDeclaration>` is always of type `float` and always has a default value in between 0.0 and 1.0.
- It is used by the response processing as follows:
  - If the final test score is greater or equal to the defined threshold value, the test is considered passed and the `FEEDBACK` `<outcomeDeclaration>` is set to `RESULT_OK`.
  - If the final test score is less than the defined threshold value, the test is considered failed and the `FEEDBACK` `<outcomeDeclaration>` is set to `RESULT_NOTOK`.

*5.2.2 Element: **<outcomeProcessing>***

5.2.2.1 General

- The `<outcomeProcessing>` section is responsible for determining the end score and the corresponding feedback.
- This profile dictates that the `<outcomeDeclaration>` must set the values of the following variables:
  - `<outcomeDeclaration>` `SCORE` must be set to the final score for this test
  - In case of feedback, the `FEEDBACK` `<outcomeDeclaration>` must be set to one of the values defined in sect. 5.2.1.3/pg. 13.
- This profile limits response processing in such a way that engines can implement it without implementing the QTI "programming language".
- For QTI items (see [NLQTI-ITEM]), this was implemented using predefined templates. Unfortunately, QTI does not allow this for tests.
- To still allow more simple rendering engines and to stay within the QTI specifications, implementation is defined as follows:
  - `outcomeProcessing` is always done using a simple standard algorithm. A rendering engine is allowed to hard code this. In other words: A rendering engine does not have to interpret the code inside the `<outcomeProcessing>` element (but is of course allowed to do so).
  - Correct QTI code for the `outcomeProcessing` must always be part of the test. This profile defines exactly how this should look like.
- The result of all this is a `<outcomeProcessing>` element with one (just score) or two (score and feedback) `<outcomeCondition>` elements:

```
<outcomeProcessing>

  <outcomeCondition>
    <!-- Fixed section for computing the score. See sect. 5.2.2.2/pg. 15 -->
  </outcomeCondition>

  <outcomeCondition>
    <!-- Optional section for taking care of the feedback.
         See sect. 5.2.2.3/pg. 16 -->
  </outcomeCondition>

</outcomeProcessing>
```

5.2.2.2 outcomeProcessing for computing the score

- For computing the score, the following code fragment must *always* be present in the `<outcomeProcessing>`:

```
<outcomeProcessing>

  <outcomeCondition>
    <outcomeIf>
      <or>
        <isNull>
          <outcomeMaximum outcomeIdentifier="SCORE" weightIdentifier="WEIGHT"/>
        </isNull>
        <equal toleranceMode="exact">
          <sum>
            <outcomeMaximum outcomeIdentifier="SCORE" weightIdentifier="WEIGHT"/>
          </sum>
          <baseValue baseType="float">0.0</baseValue>
        </equal>
      </or>
      <setOutcomeValue identifier="SCORE">
        <baseValue baseType="float">1.0</baseValue>
      </setOutcomeValue>
    </outcomeIf>
    <outcomeElse>
      <setOutcomeValue identifier="SCORE">
        <divide>
          <sum>
            <testVariables variableIdentifier="SCORE" weightIdentifier="WEIGHT"/>
          </sum>
          <sum>
            <outcomeMaximum outcomeIdentifier="SCORE" weightIdentifier="WEIGHT"/>
          </sum>
        </divide>
      </setOutcomeValue>
    </outcomeElse>
  </outcomeCondition>

</outcomeProcessing>
```

- Rendering engines are allowed to hard code the score computation. The following algorithm must be used:
  - If all items have no score or all weights are 0.0, the end score is 1.0
  - For all other cases:
    - Add up all (scores of the individual items, multiplied with their weight)
    - Divide by the sum of all weights
- Definition of the weight for items is handled in sect. 3.5.1/pg. 10
- For dynamic test (sect. 4/pg. 11): The QTI documentation is a bit vague about this, but we assume here that calculations are done only for those items actually presented to the learner.

5.2.2.3 outcomeProcessing for feedback

- A test is assumed to have feedback if `<testFeedback>` elements are present.
- Which feedback to show is guided by the `outcomeDeclaration FEEDBACK` (see sect. 5.2.1.3/pg. 13)
- In case of feedback, the following code fragment must *always* be present in the `<outcomeProcessing>`:

```
<outcomeProcessing>

  <outcomeCondition>
    <!-- Scoring. See sect. 5.2.2.2/pg. 15 -->
  </outcomeCondition>

  <outcomeCondition>
    <outcomeIf>
      <gte>
        <variable identifier="SCORE"/>
        <variable identifier="FEEDBACK_TRESHOLD"/>
      </gte>
      <setOutcomeValue identifier="FEEDBACK">
        <baseValue baseType="identifier">RESULT_OK</baseValue>
      </setOutcomeValue>
    </outcomeIf>
    <outcomeElse>
      <setOutcomeValue identifier="FEEDBACK">
        <baseValue baseType="identifier">RESULT_NOTOK</baseValue>
      </setOutcomeValue>
    </outcomeElse>
  </outcomeCondition>

</outcomeProcessing>
```

5.2.3 Elements: ***<testFeedback>***

- A test without feedback is not allowed to contain `<testFeedback>` elements.
- For a test with feedback, `<testFeedback>` elements must be present:

```
  <!-- testFeedback for RESULT_OK is optional -->
  <testFeedback outcomeIdentifier="FEEDBACK" identifier="RESULT_OK"
     showHide="show"> … </testFeedback>
  <testFeedback outcomeIdentifier="FEEDBACK" identifier="RESULT_NOTOK"
     showHide="show"> … </testFeedback>
```

- Attributes of `<testFeedback>`:

Naam	Prf?	M	Type	Remarks
access	Lim	1	string	Fixed value: "atEnd"
outcomeIdentifier	Lim	1	identifier	Fixed value: "FEEDBACK"
showHide	Lim	1	string	Fixed value: "show"
identifier	Lim	1	identifier	Limited to: "RESULT_OK" "RESULT_NOTOK"
title	No			

6 Additional elements

6.1 <timeLimits> element

The element `<timeLimits>` limits the time a learner is allowed to take for a (section of the) test. This profile limits setting time limits to the full test (element `<assessmentTest>`) or to the individual items (element `<assessmentItemRef>`). Time is expressed in seconds.

Attributes of the `<timeLimits>` element:

Name	Prf?	M	Type	Remarks
`minTime`	Yes	?	integer	The minimum number of seconds a learner must take for this test/item. Applicable only if the test is defined as linear (see `testPart's` sect. 3.2/pg. 8, attribute `navigationMode`)
`maxTime`	Yes	?	integer	The maximum number of seconds a learner can take for this test/item.

6.2 <itemSessionControl> element

The `<itemSessionControl>` element defines properties for a single item inside a test. This profile limits setting these properties to the full test (element `<assessmentTest>`) or to the individual items (element `<assessmentItemRef>`).

Attributes of the `<itemSessionControl>` element:

Name	Prf?	M	Type	Remarks
`maxAttempts`	Yes	?	integer	The allowed number of attempts to answer the item. For no limit specify 0. Default is 1. This attribute is applicable only for non-adaptive items. Since this profile does not allow adaptive items only, it's always applicable.
`showFeedback`	Yes	?	boolean	Whether or not feedback must be shown (at all). Warning: Default value is `false`, meaning no feedback will ever be shown if you don't explicitly set it to `true`. It therefore makes sense to set this to `true` on the highest level (`testPart`).
`allowReview`	Yes	?	boolean	Whether or not a review of the answers is allowed after all attempts (defined using the `maxAttempts` attribute) are exceeded. Default is `true`.
`showSolution`	Yes	?	boolean	Whether or not learner is allowed to see the answer/solution to the item. Default is `false`.
`allowComment`	Yes	?	boolean	Whether or not the learner is allowed toad comments to items. QTI does not define a default. This profile assumes a default of `false`.
`allowSkipping`	Yes	?	boolean	Whether or not items can be skipped (meaning there can be questions without a score). Default is `true`.
`validateResponses`	Yes	?	boolean	Whether or not invalid responses to items are accepted (e.g. text where a number is expected). QTI does not define a default. This profile assumes a default of `true`. Applicable only in a `testPart` for which `submissionMode="individual"`, see sect. 3.2/pg. 8.