

Schema documentation for swiML.xsd

november 28, 2024

Table of Contents

Namespace: "https://github.com/bartneck/swiML"	2
Schema(s)	2
Main schema swiML.xsd	2
Element(s)	2
Element program	2
Element program / title	5
Element program / author	6
Element program / author / firstName	6
Element program / author / lastName	7
Element program / author / email	7
Element program / programDescription	7
Element program / creationDate	8
Element program / poolLength	8
Element program / lengthUnit	8
Element program / programAlign	9
Element program / numeralSystem	9
Element program / hideIntro	9
Element program / layoutWidth	10
Element program / instruction	10
Element instructionType / segmentName	13
Element instructionType / repetition	13
Element repetitionType / repetitionCount	16
Element repetitionType / simplify	16
Element repetitionType / repetitionDescription	16
Element instructionGroup / length	17
Element lengthType / lengthAsDistance	17
Element lengthType / lengthAsTime	18
Element lengthType / lengthAsLaps	18
Element instructionGroup / stroke	18
Element strokeType / standardStroke	19
Element strokeType / kicking	19
Element kickStyle / orientation	20
Element kickStyle / legMovement	20
Element kickStyle / standardKick	21
Element strokeType / drill	21
Element drillType / drillName	22
Element drillType / drillStroke	22
Element instructionGroup / rest	23
Element restType / afterStop	24
Element restType / sinceStart	24
Element restType / sinceLastRest	24
Element restType / inOut	25
Element instructionGroup / intensity	25
Element intensityProfile / startIntensity	26
Element intensityType / percentageEffort	26
Element intensityType / zone	26
Element intensityType / percentageHeartRate	27
Element intensityProfile / stopIntensity	27
Element instructionGroup / breath	28
Element instructionGroup / underwater	28
Element instructionGroup / equipment	28
Element instructionGroup / instructionDescription	29
Element repetitionType / instruction	29
Element instructionType / pyramid	32
Element pyramidType / startLength	34
Element pyramidType / stopLength	35
Element pyramidType / increment	35
Element pyramidType / incrementLengthUnit	36
Element pyramidType / isPointy	36
Element instructionType / continue	36
Element continueType / continueLength	38
Element continueType / instruction	38

Element instructionType / excludeAlign	41
Simple Type(s)	41
Simple Type titleString	41
Simple Type emailAddress	41
Simple Type descriptionString	42
Simple Type lengthUnits	42
Simple Type numeralSystems	43
Simple Type segmentNameType	43
Simple Type instructionDescriptionType	43
Simple Type standardStrokeType	44
Simple Type orientationType	44
Simple Type legMovementType	45
Simple Type drillNameType	45
Simple Type percentType	46
Simple Type zoneType	47
Simple Type equipmentType	47
Simple Type equipmentList	47
Complex Type(s)	48
Complex Type instructionType	48
Complex Type repetitionType	52
Complex Type lengthType	56
Complex Type strokeType	56
Complex Type kickStyle	57
Complex Type drillType	57
Complex Type restType	58
Complex Type intensityProfile	59
Complex Type intensityType	59
Complex Type pyramidType	60
Complex Type continueType	62
Element Group(s)	64
Element Group instructionGroup	64

Namespace: "<https://github.com/bartneck/swiML>"

Schema(s)

Main schema **swiML.xsd**

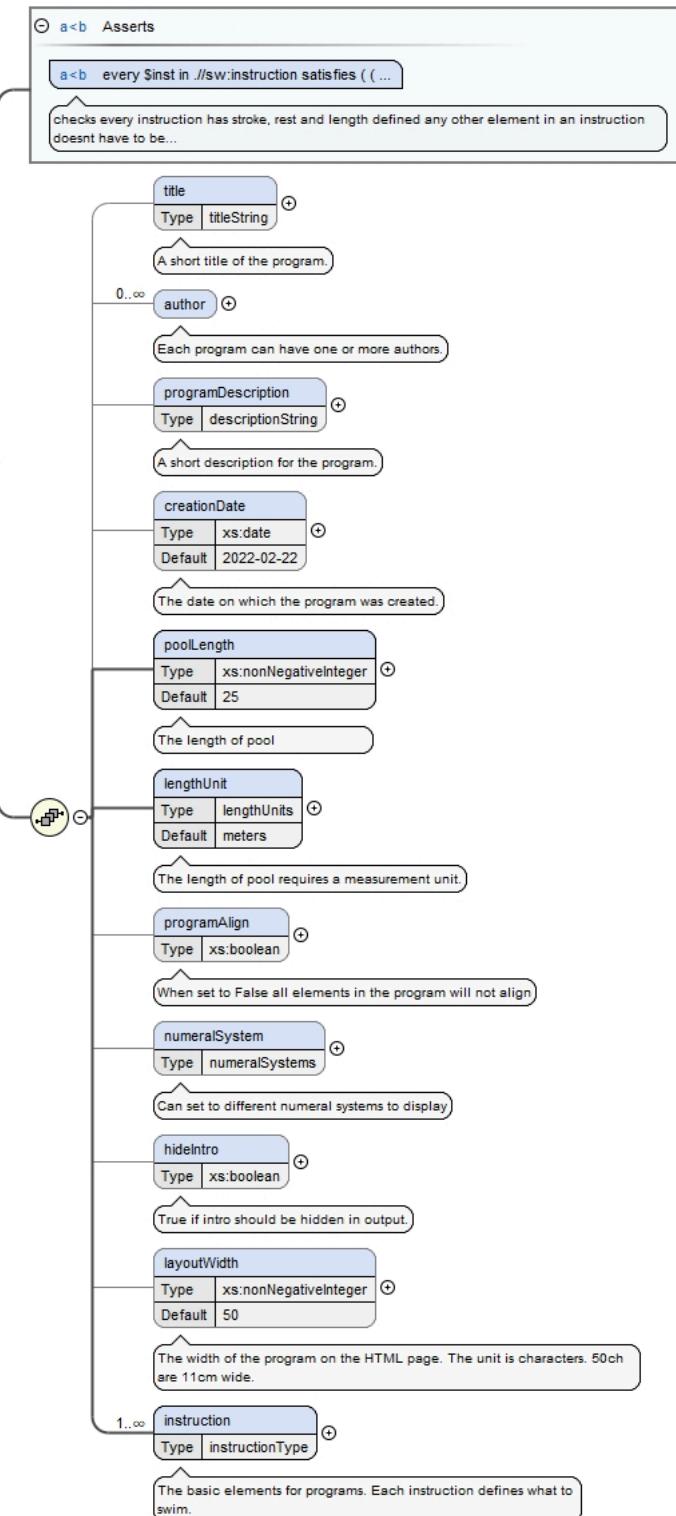
Namespace	https://github.com/bartneck/swiML
Properties	attribute form default: unqualified element form default: qualified version: 2.3

Element(s)

Element **program**

Namespace	https://github.com/bartneck/swiML

Diagram



Properties	content: complex
Model	title{0,1} , author* , programDescription{0,1} , creationDate{0,1} , poolLength , lengthUnit , programAlign{0,1} , numeralSystem{0,1} , hideIntro{0,1} , layoutWidth{0,1} , instruction+
Children	author, creationDate, hideIntro, instruction, layoutWidth, lengthUnit, numeralSystem, poolLength, programAlign, programDescription, title
Instance	<pre><program xmlns="https://github.com/bartnecke/swiML"> <title>{0,1}</title> <author>{0,unbounded}</author> <programDescription>{0,1}</programDescription> <creationDate>{0,1}</creationDate> <poolLength>{1,1}</poolLength></pre>

	<pre> <lengthUnit>{1,1}</lengthUnit> <programAlign>{0,1}</programAlign> <numeralSystem>{0,1}</numeralSystem> <hideIntro>{0,1}</hideIntro> <layoutWidth>{0,1}</layoutWidth> <instruction>{1,unbounded}</instruction> </program> </pre>	
Asserts	<p>Test</p> <p>every \$inst in ./sw:instruction satisfies ((\$inst/ancestor-or-self::*/sw:stroke and \$inst/ancestor-or-self::*/sw:length) or \$inst/sw:repetition or \$inst/sw:continue or \$inst/sw:pyramid or \$inst/sw:segmentName)</p> <p>checks every instruction has stroke, rest and length defined any other element in an instruction doesnt have to be defined</p>	XPath default namespace
Source	<pre> <xs:element name="program"> <!-- ===== --> <!-- The meta information for each program --> <!-- ===== --> <xs:complexType> <xs:sequence> <!-- The title of the program --> <xs:element name="title" type="titleString" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>A short title of the program.</xs:documentation> </xs:annotation> </xs:element> <!-- The author(s) of the program --> <xs:element maxOccurs="unbounded" minOccurs="0" name="author"> <xs:annotation> <xs:documentation>Each program can have one or more authors.</xs:documentation> </xs:annotation> <xs:complexType> <xs:sequence> <xs:element name="firstName" minOccurs="1" type="xs:string"> <xs:annotation> <xs:documentation>The first name of the author. Can contain middle names if necessary.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="lastName" minOccurs="1" type="xs:string"> <xs:annotation> <xs:documentation>The last name of the author.</xs:documentation> </xs:annotation> </xs:element> <xs:element minOccurs="0" name="email" type="emailAddress"> <xs:annotation> <xs:documentation>The email address of the author (optional).</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType> </xs:element> <!-- The description of the program --> <xs:element name="programDescription" type="descriptionString" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>A short description for the program.</xs:documentation> </xs:annotation> </xs:element> <!-- The date --> <xs:element minOccurs="0" name="creationDate" type="xs:date" maxOccurs="1" default="2022-02-22"> <xs:annotation> <xs:documentation>The date on which the program was created.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="poolLength" minOccurs="1" maxOccurs="1" type="xs:nonNegativeInteger" default="25"> <xs:annotation> <xs:documentation>The length of pool</xs:documentation> </xs:annotation> </xs:element> <xs:element name="lengthUnit" minOccurs="1" maxOccurs="1" type="lengthUnits" default="meters"> <xs:annotation> <xs:documentation>The length of pool requires a measurement unit.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="programAlign" minOccurs="0" maxOccurs="1" type="xs:boolean"> <xs:annotation> </pre>	

```

<xs:documentation>When set to False all elements in the program will not align</xs:documentation>
</xs:element>
</xs:annotation>
</xs:element>
<xs:element name="numeralSystem" minOccurs="0" maxOccurs="1" type="numeralSystems">
    <xs:annotation>
        <xs:documentation>Can set to different numeral systems to display</xs:documentation>
    </xs:annotation>
</xs:element>
<!-- Element to hide the intro text -->
<xs:element name="hideIntro" minOccurs="0" maxOccurs="1" type="xs:boolean">
    <xs:annotation>
        <xs:documentation>True if intro should be hidden in output.</xs:documentation>
    </xs:annotation>
</xs:element>
<!-- Element to set the width of the program in HTML -->
<!-- The unit is characters. 50ch are 11cm wide -->
<xs:element name="layoutWidth" minOccurs="0" maxOccurs="1" type="xs:nonNegativeInteger" default="50">
    <xs:annotation>
        <xs:documentation>The width of the program on the HTML page. The unit is characters. 50ch are 11cm wide.</xs:documentation>
    </xs:annotation>
</xs:element>
<!-- ===== -->
<!-- The main element(s) for each program. Each instruction can contain instructions. This is recursion. -->
<!-- This is the main recursive element for a program -->
<!-- ===== -->
<xs:element name="instruction" type="instructionType" minOccurs="1" maxOccurs="unbounded">
    <xs:annotation>
        <xs:documentation>The basic elements for programs. Each instruction defines what to swim.</xs:documentation>
    </xs:annotation>
    <xs:unique name="mainEquipmentUnique">
        <xs:annotation>
            <xs:documentation>Ensures all equipment values in an instruction are unique</xs:documentation>
        </xs:annotation>
        <xs:selector xpath=".//sw:equipment" />
        <xs:field xpath="./*" />
    </xs:unique>
    </xs:element>
</xs:sequence>
<xs:assert test=" every $inst in .//sw:instruction satisfies ( ($inst/ancestor-or-self::*/sw:stroke and $inst/ancestor-or-self::*/sw:length) or $inst/sw:continue or $inst/sw:segmentName ) or $inst/sw:repetition or $inst/sw:pyramid " >
    <xs:annotation>
        <xs:documentation>checks every instruction has stroke, rest and length defined any other element in an instruction doesnt have to be defined</xs:documentation>
    </xs:annotation>
</xs:assert>
</xs:complexType>
</xs:element>

```

Element program / title

Namespace	https://github.com/bartneck/swiML						
Annotations	A short title of the program.						
Diagram	<p>The diagram shows a class named 'title' with two associations. One association is labeled 'Type' and the other is labeled 'titleString'. A constraint box is connected to the 'titleString' association with the text 'The length of the title is constraint in length.' A callout box below the 'titleString' association specifies 'A short title of the program.'</p>						
Type	titleString						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Facets	maxLength 60						
Source	<pre> <xs:element name="title" type="titleString" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>A short title of the program.</xs:documentation> </xs:annotation> </pre>						

<code></xs:element></code>

Element program / author

Namespace	https://github.com/bartneck/swiML						
Annotations	Each program can have one or more authors.						
Diagram	<pre> classDiagram class author { <<Each program can have one or more authors.>> +firstName : xs:string +lastName : xs:string +email : emailAddress } author < --> xs:string author < --> emailAddress </pre> <p>The diagram illustrates the structure of the 'author' element. It consists of three attributes: 'firstName' (Type: xs:string), 'lastName' (Type: xs:string), and 'email' (Type: emailAddress). The 'firstName' and 'lastName' attributes are marked with a circled plus sign (+) indicating they are required. The 'email' attribute is marked with a circled plus sign (+) and has a note below it stating 'The email address of the author (optional)'.</p>						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>unbounded</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	unbounded
content:	complex						
minOccurs:	0						
maxOccurs:	unbounded						
Model	firstName , lastName , email{0,1}						
Children	email, firstName, lastName						
Instance	<pre> <author xmlns="https://github.com/bartneck/swiML"> <firstName>{1,1}</firstName> <lastName>{1,1}</lastName> <email>{0,1}</email> </author> </pre>						
Source	<pre> <xs:element maxOccurs="unbounded" minOccurs="0" name="author"> <xs:annotation> <xs:documentation>Each program can have one or more authors.</xs:documentation> </xs:annotation> <xs:complexType> <xs:sequence> <xs:element name="firstName" minOccurs="1" type="xs:string"> <xs:annotation> <xs:documentation>The first name of the author. Can contain middle names if necessary.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="lastName" minOccurs="1" type="xs:string"> <xs:annotation> <xs:documentation>The last name of the author.</xs:documentation> </xs:annotation> </xs:element> <xs:element minOccurs="0" name="email" type="emailAddress"> <xs:annotation> <xs:documentation>The email address of the author (optional).</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:complexType> </xs:element> </pre>						

Element program / author / firstName

Namespace	https://github.com/bartneck/swiML
Annotations	The first name of the author. Can contain middle names if necessary.
Diagram	<pre> classDiagram class firstName { <<The first name of the author. Can contain middle names if necessary.>> <<Built-in primitive type. The string datatype represents character strings in XML.>> +xs:string } firstName < --> xs:string </pre> <p>The diagram shows the 'firstName' attribute with its type 'xs:string'. A note below the attribute states 'The first name of the author. Can contain middle names if necessary.' and another note next to the type indicates 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xs:string

Properties	content: simple minOccurs: 1
Source	<pre><xs:element name="firstName" minOccurs="1" type="xs:string"> <xs:annotation> <xs:documentation>The first name of the author. Can contain middle names if necessary.</xs:documentation> </xs:annotation> </xs:element></pre>

Element program / author / lastName

Namespace	https://github.com/bartneck/swiML
Annotations	The last name of the author.
Diagram	<p>The diagram shows a class named 'lastName' with a multiplicity of 1. It is associated with a type 'xs:string'. A note below the association line states: 'The last name of the author.' Another note next to the type symbol states: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	xs:string
Properties	content: simple minOccurs: 1
Source	<pre><xs:element name="lastName" minOccurs="1" type="xs:string"> <xs:annotation> <xs:documentation>The last name of the author.</xs:documentation> </xs:annotation> </xs:element></pre>

Element program / author / email

Namespace	https://github.com/bartneck/swiML
Annotations	The email address of the author (optional).
Diagram	<p>The diagram shows a class named 'email' with a multiplicity of 0..*. It is associated with a type 'emailAddress'. A note below the association line states: 'The email address of the author (optional).' Another note next to the type symbol states: 'The pattern checks for valid email addresses.'</p>
Type	emailAddress
Properties	content: simple minOccurs: 0
Facets	pattern [^@]+@[^\.\]+\.\.+
Source	<pre><xs:element minOccurs="0" name="email" type="emailAddress"> <xs:annotation> <xs:documentation>The email address of the author (optional).</xs:documentation> </xs:annotation> </xs:element></pre>

Element program / programDescription

Namespace	https://github.com/bartneck/swiML
Annotations	A short description for the program.
Diagram	<p>The diagram shows a class named 'programDescription' with a multiplicity of 0..*. It is associated with a type 'descriptionString'. A note below the association line states: 'A short description for the program.' Another note next to the type symbol states: 'The length of the description text is constraint in length.'</p>
Type	descriptionString
Properties	content: simple minOccurs: 0 maxOccurs: 1

Facets	maxLength	400
Source	<pre><xs:element name="programDescription" type="descriptionString" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>A short description for the program.</xs:documentation> </xs:annotation> </xs:element></pre>	

Element program / creationDate

Namespace	https://github.com/bartneck/swiML									
Annotations	The date on which the program was created.									
Diagram	<p>The diagram shows a class named 'creationDate' with a single attribute 'Type' set to 'xs:date'. A note below the class states: 'The date on which the program was created.' A callout points to the 'xs:date' icon with the text: 'Built-in primitive type. The date datatype represents a calendar date.'</p>									
Type	xs:date									
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>default:</td> <td>2022-02-22</td> </tr> </table>		content:	simple	minOccurs:	0	maxOccurs:	1	default:	2022-02-22
content:	simple									
minOccurs:	0									
maxOccurs:	1									
default:	2022-02-22									
Source	<pre><xs:element minOccurs="0" name="creationDate" type="xs:date" maxOccurs="1" default="2022-02-22"> <xs:annotation> <xs:documentation>The date on which the program was created.</xs:documentation> </xs:annotation> </xs:element></pre>									

Element program / poolLength

Namespace	https://github.com/bartneck/swiML									
Annotations	The length of pool									
Diagram	<p>The diagram shows a class named 'poolLength' with a single attribute 'Type' set to 'xs:nonNegativeInteger'. A note below the class states: 'The length of pool'. A callout points to the 'xs:nonNegativeInteger' icon with the text: 'Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to...'</p>									
Type	xs:nonNegativeInteger									
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>default:</td> <td>25</td> </tr> </table>		content:	simple	minOccurs:	1	maxOccurs:	1	default:	25
content:	simple									
minOccurs:	1									
maxOccurs:	1									
default:	25									
Source	<pre><xs:element name="poolLength" minOccurs="1" maxOccurs="1" type="xs:nonNegativeInteger" default="25"> <xs:annotation> <xs:documentation>The length of pool</xs:documentation> </xs:annotation> </xs:element></pre>									

Element program / lengthUnit

Namespace	https://github.com/bartneck/swiML	
Annotations	The length of pool requires a measurement unit.	
Diagram	<p>The diagram shows a class named 'lengthUnit' with a single attribute 'Type' set to 'lengthUnits'. A note below the class states: 'The length of pool requires a measurement unit.'. A callout points to the 'lengthUnits' icon with the text: 'The unit of measurement for the length of the target pool (meter or yards).'</p>	
Type	lengthUnits	

Properties	content: simple minOccurs: 1 maxOccurs: 1 default: meters
Facets	enumeration meters enumeration kilometers enumeration miles enumeration yards
Source	<pre><xs:element name="lengthUnit" minOccurs="1" maxOccurs="1" type="lengthUnits" default="meters"> <xs:annotation> <xs:documentation>The length of pool requires a measurement unit.</xs:documentation> </xs:annotation> </xs:element></pre>

Element program / programAlign

Namespace	https://github.com/bartneck/swiML
Annotations	When set to False all elements in the program will not align
Diagram	<p>programAlign</p> <p>Type xs:boolean</p> <p>When set to False all elements in the program will not align.</p> <p>xs:boolean</p> <p>Built-in primitive type. It defines the boolean values true and false.</p>
Type	xs:boolean
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><xs:element name="programAlign" minOccurs="0" maxOccurs="1" type="xs:boolean"> <xs:annotation> <xs:documentation>When set to False all elements in the program will not align</xs:documentation> </xs:annotation> </xs:element></pre>

Element program / numeralSystem

Namespace	https://github.com/bartneck/swiML
Annotations	Can set to different numeral systems to display
Diagram	<p>numeralSystem</p> <p>Type numeralSystems</p> <p>Can set to different numeral systems to display.</p> <p>numeralSystems</p> <p>Numeral system type to be used for displaying the document</p>
Type	numeralSystems
Properties	content: simple minOccurs: 0 maxOccurs: 1
Facets	enumeration decimal enumeration roman
Source	<pre><xs:element name="numeralSystem" minOccurs="0" maxOccurs="1" type="numeralSystems"> <xs:annotation> <xs:documentation>Can set to different numeral systems to display</xs:documentation> </xs:annotation> </xs:element></pre>

Element program / hideIntro

Namespace	https://github.com/bartneck/swiML
-----------	-----------------------------------

Annotations	True if intro should be hidden in output.						
Diagram	<p>Diagram illustrating the type xs:boolean:</p> <pre> classDiagram class hideIntro { Type xs:boolean } xs:boolean < -- xs:boolean xs:boolean --> BuiltInPrimitiveType: "Built-in primitive type. It defines the boolean values true and false." </pre>						
Type	xs:boolean						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<pre> <xss:element name="hideIntro" minOccurs="0" maxOccurs="1" type="xs:boolean"> <xss:annotation> <xss:documentation>True if intro should be hidden in output.</xss:documentation> </xss:annotation> </xss:element> </pre>						

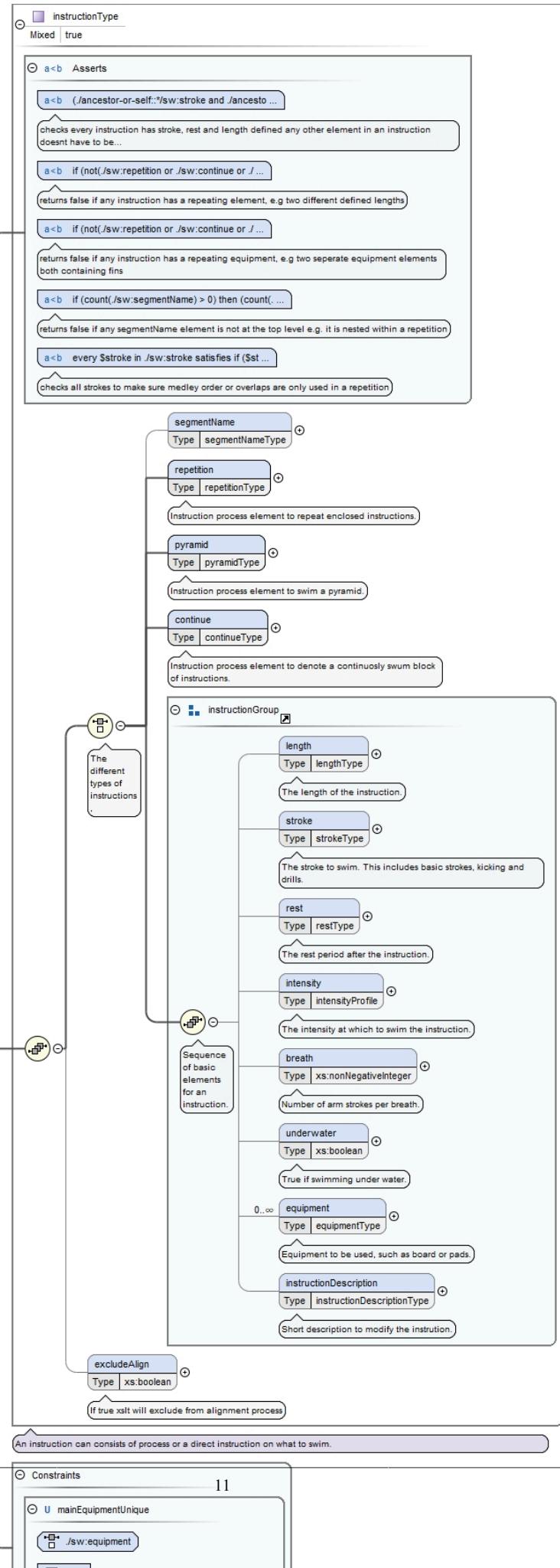
Element program / layoutWidth

Namespace	https://github.com/bartneck/swiML								
Annotations	The width of the program on the HTML page. The unit is characters. 50ch are 11cm wide.								
Diagram	<p>Diagram illustrating the type xs:nonNegativeInteger:</p> <pre> classDiagram class layoutWidth { Type xs:nonNegativeInteger Default 50 } xs:nonNegativeInteger < -- xs:nonNegativeInteger xs:nonNegativeInteger --> BuiltInDerivedType: "Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to..." </pre>								
Type	xs:nonNegativeInteger								
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>default:</td> <td>50</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1	default:	50
content:	simple								
minOccurs:	0								
maxOccurs:	1								
default:	50								
Source	<pre> <xss:element name="layoutWidth" minOccurs="0" maxOccurs="1" type="xs:nonNegativeInteger" default="50"> <xss:annotation> <xss:documentation>The width of the program on the HTML page. The unit is characters. 50ch are 11cm wide.</xss:documentation> </xss:annotation> </xss:element> </pre>								

Element program / instruction

Namespace	https://github.com/bartneck/swiML
Annotations	The basic elements for programs. Each instruction defines what to swim.

Diagram



Type	instructionType	
Properties	content:	complex
	minOccurs:	1
	maxOccurs:	unbounded
	mixed:	true
Model	(segmentName{0,1} repetition pyramid continue (length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1})) , excludeAlign{0,1}	
Children	breath, continue, equipment, excludeAlign, instructionDescription, intensity, length, pyramid, repetition, rest, segmentName, stroke, underwater	
Instance	<pre><instruction xmlns="https://github.com/bartneck/swiML"> <segmentName>{0,1}</segmentName> <repetition>{1,1}</repetition> <pyramid>{1,1}</pyramid> <continue>{1,1}</continue> <length>{0,1}</length> <stroke>{0,1}</stroke> <rest>{0,1}</rest> <intensity>{0,1}</intensity> <breath>{0,1}</breath> <underwater>{0,1}</underwater> <equipment>{0,unbounded}</equipment> <instructionDescription>{0,1}</instructionDescription> <excludeAlign>{0,1}</excludeAlign> </instruction></pre>	
Asserts	<p>Test</p> <pre>(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName</pre> <p>checks every instruction has stroke, rest and length defined any other element in an instruction doesnt have to be defined</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in ./* satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'length' or name() = 'stroke' or name() = 'rest' or name() = 'intensity' or name() = 'breath' or name() = 'underwater'] satisfies not(name(\$element) = name(\$match))) else (true())</pre> <p>returns false if any instruction has a repeating element, e.g two different defined lengths</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in ./*[name() = 'equipment'] satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'equipment'] satisfies not(\$element/text() = \$match/text())) else (true())</pre> <p>returns false if any instruction has a repeating equipment, e.g two seperate equipment elements both containing fins</p> <pre>if(count(.//sw:segmentName) > 0) then (count(.//sw:segmentName/../../ancestor::*) = 0) else (true())</pre> <p>returns false if any segmentName element is not at the top level e.g. it is nested within a repetition</p> <pre>every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOverlap' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'reverseIndividualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOverlap' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'reverseIndividualMedleyOrder') then (\$stroke/ancestor::*/sw:repetition) or (\$stroke/ancestor::*/sw:continue/sw:continueLength) else (\$stroke/parent::*)</pre>	<p>XPath default namespace</p>

	Test		XPath default namespace		
	checks all strokes to make sure medley order or overlaps are only used in a repetition				
Identity constraints	QName	Type	Refer	Selector	Field(s)
	mainEquipmentUnique	unique		./sw:equipment	.
Source	<pre><xs:element name="instruction" type="instructionType" minOccurs="1" maxOccurs="unbounded"> <xs:annotation> <xs:documentation>The basic elements for programs. Each instruction defines what to swim.</xs:documentation> </xs:annotation> <xs:unique name="mainEquipmentUnique"> <xs:annotation> <xs:documentation>Ensures all equipment values in an instruction are unique</xs:documentation> </xs:annotation> <xs:selector xpath=".//sw:equipment" /> <xs:field xpath=".//." /> </xs:unique> </xs:element></pre>				

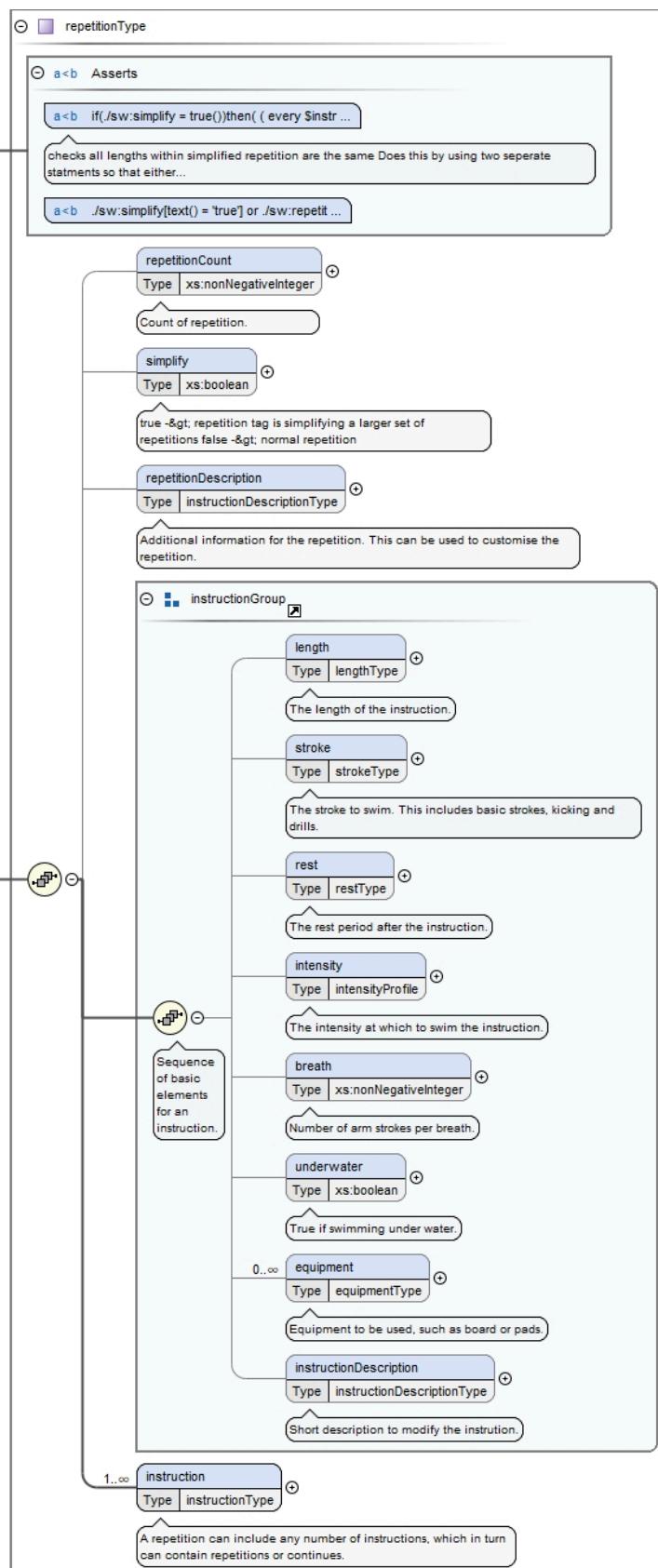
Element instructionType / segmentName

Namespace	https://github.com/bartneck/swiML						
Diagram							
Type	segmentNameType						
Properties	<table> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Facets	maxLength 50						
Source	<pre><xs:element name="segmentName" minOccurs="0" maxOccurs="1" type="segmentNameType" /></pre>						

Element instructionType / repetition

Namespace	https://github.com/bartneck/swiML
Annotations	Instruction process element to repeat enclosed instructions.

Diagram

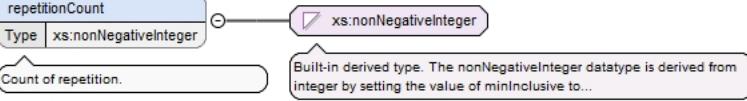


Type	repetitionType
Properties	content: complex
Model	repetitionCount{0,1} , simplify{0,1} , repetitionDescription{0,1} , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1} , instruction+

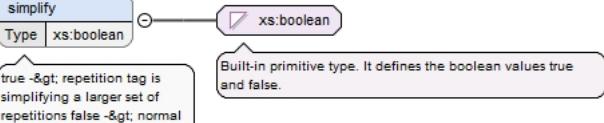
Children	breath, equipment, instruction, instructionDescription, intensity, length, repetitionCount, repetitionDescription, rest, simplify, stroke, underwater	
Instance	<pre> <repetition xmlns="https://github.com/bartneck/swiML"> <repetitionCount>{0,1}</repetitionCount> <simplify>{0,1}</simplify> <repetitionDescription>{0,1}</repetitionDescription> <length>{0,1}</length> <stroke>{0,1}</stroke> <rest>{0,1}</rest> <intensity>{0,1}</intensity> <breath>{0,1}</breath> <underwater>{0,1}</underwater> <equipment>{0,unbounded}</equipment> <instructionDescription>{0,1}</instructionDescription> <instruction>{1,unbounded}</instruction> </repetition></pre>	
Asserts	<p>Test</p> <pre> if(.//sw:simplify = true())then((every \$instruction in ./sw:instruction[not(.//sw:pyramid or ./sw:segmentName)] satisfies((if(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) then(if(count(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) = 1) then(number((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1]//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsDistance)) else(sum((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1]//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsDistance)) else(0)+(if(\$instruction/descendant-or-self::sw:continueLength) then(number(\$instruction/descendant-or-self::sw:continueLength) else(0)) = number(((./descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1]//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsDistance) (./descendant-or-self::sw:continueLength)[1])) or(every \$instruction in ./sw:instruction[not(.//sw:pyramid or ./sw:segmentName)] satisfies((if(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) then(if(count(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) = 1) then(number((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1]//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsLaps)) else(sum((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1]//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsLaps)) else(0)+(if(\$instruction/descendant-or-self::sw:continueLength) then(number(\$instruction/descendant-or-self::sw:continueLength) else(0)) = number(((./descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1]//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsLaps) (./descendant-or-self::sw:continueLength)[1]))) else(true())</pre> <p>checks all lengths within simplified repetition are the same Does this by using two separate statements so that either all lengthAsDistances are the same or all lengthAsLaps checks that every instruction, repetition and continue has the same base length e.g. 100m , 4x100m and 50,50 swim as 100 ; these are all of length 100 so can be simplified in the same repetition has to choose whether to use continueLength or add up all instructions within a continue</p>	

	Test ./sw:simplify[text() = 'true'] or ./sw:repetitionCount and not(./sw:simplify[text() = 'true'] and ./sw:repetitionCount)	XPath default namespace
Source	<pre><xs:element name="repetition" type="repetitionType"> <xs:annotation> <xs:documentation>Instruction process element to repeat enclosed instructions.</xs:documentation> </xs:annotation> </xs:element></pre>	

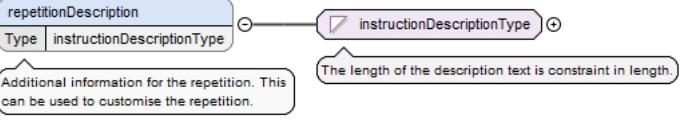
Element repetitionType / repetitionCount

Namespace	https://github.com/bartneck/swiML						
Annotations	Count of repetition.						
Diagram							
Type	xs:nonNegativeInteger						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<pre><xs:element name="repetitionCount" type="xs:nonNegativeInteger" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>Count of repetition.</xs:documentation> </xs:annotation> </xs:element></pre>						

Element repetitionType / simplify

Namespace	https://github.com/bartneck/swiML						
Annotations	true -> repetition tag is simplifying a larger set of repetitions false -> normal repetition						
Diagram							
Type	xs:boolean						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<pre><xs:element name="simplify" minOccurs="0" maxOccurs="1" type="xs:boolean"> <xs:annotation> <xs:documentation>true -> repetition tag is simplifying a larger set of repetitions false -> normal repetition</xs:documentation> </xs:annotation> </xs:element></pre>						

Element repetitionType / repetitionDescription

Namespace	https://github.com/bartneck/swiML		
Annotations	Additional information for the repetition. This can be used to customise the repetition.		
Diagram			
Type	instructionDescriptionType		
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> </table>	content:	simple
content:	simple		

	minOccurs:	0
	maxOccurs:	1
Facets	maxLength	100
Source	<pre><xs:element name="repetitionDescription" minOccurs="0" maxOccurs="1" type="instructionDescriptionType"> <xs:annotation> <xs:documentation>Additional information for the repetition. This can be used to customise the repetition.</xs:documentation> </xs:annotation> </xs:element></pre>	

Element instructionGroup / length

Namespace	https://github.com/bartneck/swiML								
Annotations	The length of the instruction.								
Diagram	<pre> classDiagram class length { <<lengthType>> <<Mixed true>> } class lengthAsDistance { <<xs:nonNegativeInteger>> } class lengthAsTime { <<xs:duration>> } class lengthAsLaps { <<xs:nonNegativeInteger>> } length "0..1" -- "0..1" lengthAsDistance length "0..1" -- "0..1" lengthAsTime length "0..1" -- "0..1" lengthAsLaps </pre> <p>The length for a swimming instruction.</p>								
Type	lengthType								
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>mixed:</td> <td>true</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1	mixed:	true
content:	complex								
minOccurs:	0								
maxOccurs:	1								
mixed:	true								
Model	lengthAsDistance lengthAsTime lengthAsLaps								
Children	lengthAsDistance, lengthAsLaps, lengthAsTime								
Instance	<pre> <length xmlns="https://github.com/bartneck/swiML"> <lengthAsDistance>{1,1}</lengthAsDistance> <lengthAsTime>{1,1}</lengthAsTime> <lengthAsLaps>{1,1}</lengthAsLaps> </length> </pre>								
Source	<pre><xs:element name="length" minOccurs="0" maxOccurs="1" type="lengthType"> <xs:annotation> <xs:documentation>The length of the instruction.</xs:documentation> </xs:annotation> </xs:element></pre>								

Element lengthType / lengthAsDistance

Namespace	https://github.com/bartneck/swiML		
Annotations	Length of instruction as distance.		
Diagram	<pre> classDiagram class lengthAsDistance { <<xs:nonNegativeInteger>> } class xsnonNegativeInteger { <<xs:nonNegativeInteger>> } lengthAsDistance "0..1" -- "0..1" xsnonNegativeInteger </pre> <p>Length of instruction as distance.</p> <p>Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to...</p>		
Type	xs:nonNegativeInteger		
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> </table>	content:	simple
content:	simple		
Source	<pre><xs:element name="lengthAsDistance" type="xs:nonNegativeInteger"></pre>		

```

<xs:annotation>
  <xs:documentation>Length of instruction as distance.</xs:documentation>
</xs:annotation>
</xs:element>

```

Element lengthType / lengthAsTime

Namespace	https://github.com/bartneck/swiML
Annotations	Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30.
Diagram	<p>Built-in primitive type. The duration datatype represents a duration of time.</p>
Type	xs:duration
Properties	content: simple
Source	<pre> <xs:element name="lengthAsTime" type="xs:duration"> <xs:annotation> <xs:documentation>Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30.</xs:documentation> </xs:annotation> </xs:element> </pre>

Element lengthType / lengthAsLaps

Namespace	https://github.com/bartneck/swiML
Annotations	Length of instruction in number of laps.
Diagram	<p>Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to...</p>
Type	xs:nonNegativeInteger
Properties	content: simple
Source	<pre> <xs:element name="lengthAsLaps" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>Length of instruction in number of laps.</xs:documentation> </xs:annotation> </xs:element> </pre>

Element instructionGroup / stroke

Namespace	https://github.com/bartneck/swiML								
Annotations	The stroke to swim. This includes basic strokes, kicking and drills.								
Diagram	<p>Stroke types.</p>								
Type	strokeType								
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>mixed:</td> <td>true</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1	mixed:	true
content:	complex								
minOccurs:	0								
maxOccurs:	1								
mixed:	true								

Model	standardStroke kicking drill
Children	drill, kicking, standardStroke
Instance	<pre><stroke xmlns="https://github.com/bartneck/swiML"> <standardStroke>{1,1}</standardStroke> <kicking>{1,1}</kicking> <drill>{1,1}</drill> </stroke></pre>
Source	<pre><xs:element name="stroke" minOccurs="0" maxOccurs="1" type="strokeType"> <xs:annotation> <xs:documentation>The stroke to swim. This includes basic strokes, kicking and drills.</xs:documentation> </xs:annotation> </xs:element></pre>

Element strokeType / standardStroke

Namespace	https://github.com/bartneck/swiML
Diagram	<pre> classDiagram class standardStroke { <<Type standardStrokeType>> } class standardStrokeType { <<Type standardStrokeType>> } standardStroke < -- standardStrokeType </pre>
Type	standardStrokeType
Properties	content: simple
Facets	enumeration butterfly enumeration backstroke enumeration breaststroke enumeration freestyle enumeration individualMedley enumeration reverseIndividualMedley enumeration individualMedleyOverlap enumeration individualMedleyOrder enumeration reverseIndividualMedley-Order enumeration any enumeration nr1 enumeration nr2 enumeration nr3 enumeration nr4 enumeration notButterfly enumeration notBackstroke enumeration notBreaststroke enumeration notFreestyle
Source	<pre><xs:element name="standardStroke" type="standardStrokeType" /></pre>

Element strokeType / kicking

Namespace	https://github.com/bartneck/swiML
Diagram	<pre> classDiagram class kicking { <<Type kickStyle>> } class kickStyle { <<Type kickStyle>> } kicking < -- kickStyle class orientation { <<Type orientationType>> } class legMovement { <<Type legMovementType>> } class standardKick { <<Type standardStrokeType>> } kickStyle < -- orientation kickStyle < -- legMovement kickStyle < -- standardKick </pre> <p>The orientation of the swimmers body.</p> <p>The style of the leg movements.</p>

Type	kickStyle
Properties	content: complex
Model	(orientation{0,1} , legMovement) standardKick
Children	legMovement, orientation, standardKick
Instance	<kicking xmlns="https://github.com/bartneck/swiML"> <orientation>{0,1}</orientation> <legMovement>{1,1}</legMovement> <standardKick>{1,1}</standardKick> </kicking>
Source	<xs:element name="kicking" type="kickStyle" />

Element kickStyle / orientation

Namespace	https://github.com/bartneck/swiML														
Annotations	The orientation of the swimmers body.														
Diagram															
Type	orientationType														
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1								
content:	simple														
minOccurs:	0														
maxOccurs:	1														
Facets	<table border="1"> <tr> <td>enumeration</td> <td>front</td> </tr> <tr> <td>enumeration</td> <td>back</td> </tr> <tr> <td>enumeration</td> <td>left</td> </tr> <tr> <td>enumeration</td> <td>right</td> </tr> <tr> <td>enumeration</td> <td>side</td> </tr> <tr> <td>enumeration</td> <td>vertical</td> </tr> <tr> <td>enumeration</td> <td>waka</td> </tr> </table>	enumeration	front	enumeration	back	enumeration	left	enumeration	right	enumeration	side	enumeration	vertical	enumeration	waka
enumeration	front														
enumeration	back														
enumeration	left														
enumeration	right														
enumeration	side														
enumeration	vertical														
enumeration	waka														
Source	<xs:element name="orientation" type="orientationType" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>The orientation of the swimmers body.</xs:documentation> <br < xs:annotation><br=""></br <> </xs:element>														

Element kickStyle / legMovement

Namespace	https://github.com/bartneck/swiML						
Annotations	The style of the leg movements.						
Diagram							
Type	legMovementType						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	1	maxOccurs:	1
content:	simple						
minOccurs:	1						
maxOccurs:	1						
Facets	<table border="1"> <tr> <td>enumeration</td> <td>flutter</td> </tr> <tr> <td>enumeration</td> <td>dolphin</td> </tr> <tr> <td>enumeration</td> <td>scissor</td> </tr> </table>	enumeration	flutter	enumeration	dolphin	enumeration	scissor
enumeration	flutter						
enumeration	dolphin						
enumeration	scissor						
Source	<xs:element name="legMovement" type="legMovementType" minOccurs="1" maxOccurs="1"> <xs:annotation> <xs:documentation>The style of the leg movements.</xs:documentation> <br < xs:annotation><br=""></br <> </xs:element>						

<code></xs:element></code>

Element `kickStyle / standardKick`

Namespace	https://github.com/bartneck/swiML
Diagram	<pre> classDiagram class standardKick { <<standardStrokeType>> } standardKick < -- standardStrokeType </pre>
Type	standardStrokeType
Properties	<p>content: simple</p> <p>minOccurs: 1</p> <p>maxOccurs: 1</p>
Facets	<p>enumeration butterfly</p> <p>enumeration backstroke</p> <p>enumeration breaststroke</p> <p>enumeration freestyle</p> <p>enumeration individualMedley</p> <p>enumeration reverseIndividualMedley</p> <p>enumeration individualMedleyOverlap</p> <p>enumeration individualMedleyOrder</p> <p>enumeration reverseIndividualMedley-Order</p> <p>enumeration any</p> <p>enumeration nr1</p> <p>enumeration nr2</p> <p>enumeration nr3</p> <p>enumeration nr4</p> <p>enumeration notButterfly</p> <p>enumeration notBackstroke</p> <p>enumeration notBreaststroke</p> <p>enumeration notFreestyle</p>
Source	<code><xs:element name="standardKick" minOccurs="1" maxOccurs="1" type="standardStrokeType"/></code>

Element `strokeType / drill`

Namespace	https://github.com/bartneck/swiML
Diagram	<pre> classDiagram class drill { <<drillType>> } drill < -- drillType class drillType { drillName drillStroke } drillName < -- drillNameType drillStroke < -- standardStrokeType note over drillType: Drills are based on stroke types. For example, the drill 123 can be swum with freestyle or backstroke. note over drillType: Drill type consists of a drill name and a stroke. For example, this could mean 6 kick drill freestyle. </pre>
Type	drillType
Properties	content: complex
Model	drillName{0,1} , drillStroke
Children	drillName, drillStroke
Instance	<code><drill xmlns="https://github.com/bartneck/swiML"> <drillName>{0,1}</drillName> <drillStroke>{1,1}</drillStroke></code>

	</drill>
Source	<xss:element name="drill" type="drillType"/>

Element `drillType / drillName`

Namespace	https://github.com/bartneck/swiML																																										
Diagram	<pre> classDiagram class drillName { <<Type drillNameType>> } class drillNameType { <<Drill names.>> } drillName "0..1" -- "1..1" drillNameType </pre>																																										
Type	drillNameType																																										
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1																																				
content:	simple																																										
minOccurs:	0																																										
maxOccurs:	1																																										
Facets	<table border="1"> <tr><td>enumeration</td><td>6KickDrill</td></tr> <tr><td>enumeration</td><td>8KickDrill</td></tr> <tr><td>enumeration</td><td>10KickDrill</td></tr> <tr><td>enumeration</td><td>12KickDrill</td></tr> <tr><td>enumeration</td><td>fingerTrails</td></tr> <tr><td>enumeration</td><td>123</td></tr> <tr><td>enumeration</td><td>bigDog</td></tr> <tr><td>enumeration</td><td>scull</td></tr> <tr><td>enumeration</td><td>singleArm</td></tr> <tr><td>enumeration</td><td>any</td></tr> <tr><td>enumeration</td><td>technic</td></tr> <tr><td>enumeration</td><td>dogPaddle</td></tr> <tr><td>enumeration</td><td>tarzan</td></tr> <tr><td>enumeration</td><td>fist</td></tr> <tr><td>enumeration</td><td>2Kick1Pull</td></tr> <tr><td>enumeration</td><td>3Kick1Pull</td></tr> <tr><td>enumeration</td><td>2Pull1Kick</td></tr> <tr><td>enumeration</td><td>3Pull1Kick</td></tr> <tr><td>enumeration</td><td>3Right3Left</td></tr> <tr><td>enumeration</td><td>2Right2Left</td></tr> <tr><td>enumeration</td><td>other</td></tr> </table>	enumeration	6KickDrill	enumeration	8KickDrill	enumeration	10KickDrill	enumeration	12KickDrill	enumeration	fingerTrails	enumeration	123	enumeration	bigDog	enumeration	scull	enumeration	singleArm	enumeration	any	enumeration	technic	enumeration	dogPaddle	enumeration	tarzan	enumeration	fist	enumeration	2Kick1Pull	enumeration	3Kick1Pull	enumeration	2Pull1Kick	enumeration	3Pull1Kick	enumeration	3Right3Left	enumeration	2Right2Left	enumeration	other
enumeration	6KickDrill																																										
enumeration	8KickDrill																																										
enumeration	10KickDrill																																										
enumeration	12KickDrill																																										
enumeration	fingerTrails																																										
enumeration	123																																										
enumeration	bigDog																																										
enumeration	scull																																										
enumeration	singleArm																																										
enumeration	any																																										
enumeration	technic																																										
enumeration	dogPaddle																																										
enumeration	tarzan																																										
enumeration	fist																																										
enumeration	2Kick1Pull																																										
enumeration	3Kick1Pull																																										
enumeration	2Pull1Kick																																										
enumeration	3Pull1Kick																																										
enumeration	3Right3Left																																										
enumeration	2Right2Left																																										
enumeration	other																																										
Source	<xss:element name="drillName" minOccurs="0" maxOccurs="1" type="drillNameType"/>																																										

Element `drillType / drillStroke`

Namespace	https://github.com/bartneck/swiML						
Annotations	Drills are based on stroke types. For example, the drill 123 can be swum with freestyle or backstroke.						
Diagram	<pre> classDiagram class drillStroke { <<Type standardStrokeType>> } class standardStrokeType { <<Drills are based on stroke types. For example, the drill 123 can be swum with freestyle or backstroke.>> } drillStroke "0..1" -- "1..1" standardStrokeType </pre>						
Type	standardStrokeType						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	1	maxOccurs:	1
content:	simple						
minOccurs:	1						
maxOccurs:	1						
Facets	enumeration butterfly						

	enumeration	backstroke
	enumeration	breaststroke
	enumeration	freestyle
	enumeration	individualMedley
	enumeration	reverseIndividualMedley
	enumeration	individualMedleyOverlap
	enumeration	individualMedleyOrder
	enumeration	reverseIndividualMedley-Order
	enumeration	any
	enumeration	nr1
	enumeration	nr2
	enumeration	nr3
	enumeration	nr4
	enumeration	notButterfly
	enumeration	notBackstroke
	enumeration	notBreaststroke
	enumeration	notFreestyle
Source	<pre><xss:element name="drillStroke" type="standardStrokeType" maxOccurs="1" minOccurs="1"> <xss:annotation> <xss:documentation>Drills are based on stroke types. For example, the drill 123 can be swum with freestyle or backstroke.</xss:documentation> </xss:annotation> </xss:element></pre>	

Element instructionGroup / rest

Namespace	https://github.com/bartneck/swiML								
Annotations	The rest period after the instruction.								
Diagram	<p>The diagram illustrates the structure of the 'rest' element. It is defined as a mixed type element ('Mixed true'). It has four associations:</p> <ul style="list-style-type: none"> afterStop: Type xs:duration. Description: Duration of rest after stopping a swimming instruction. Example: 20 seconds means that the swimmer will rest for 20... sinceStart: Type xs:duration. Description: The interval on which swimming instructions start. Example: on 1:30 means that the next instructions starts after 1:30... sinceLastRest: Type xs:duration. Description: The time since the end of the last rest. This is useful when several instructions without a rest period are swum,... inOut: Type xs:nonNegativeInteger. Description: Number of swimmers arriving. Example: 3rd in: Once the 3rd swimmer in the lane arrives, the 1st swimmer starts. <p>A note at the bottom states: The length units for a rest after a swimming instruction.</p>								
Type	restType								
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>mixed:</td> <td>true</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1	mixed:	true
content:	complex								
minOccurs:	0								
maxOccurs:	1								
mixed:	true								

Model	afterStop sinceStart sinceLastRest inOut
Children	afterStop, inOut, sinceLastRest, sinceStart
Instance	<pre><rest xmlns="https://github.com/bartneck/swiML"> <afterStop>{1,1}</afterStop> <sinceStart>{1,1}</sinceStart> <sinceLastRest>{1,1}</sinceLastRest> <inOut>{1,1}</inOut> </rest></pre>
Source	<pre><xss:element name="rest" minOccurs="0" maxOccurs="1" type="restType"> <xss:annotation> <xss:documentation>The rest period after the instruction.</xss:documentation> </xss:annotation> </xss:element></pre>

Element restType / afterStop

Namespace	https://github.com/bartneck/swiML
Annotations	Duration of rest after stopping a swimming instruction. Example: 20 seconds means that the swimmer will rest for 20 seconds after stopping the current instructions.
Diagram	
Type	xs:duration
Properties	content: simple
Source	<pre><xss:element name="afterStop" type="xs:duration"> <xss:annotation> <xss:documentation>Duration of rest after stopping a swimming instruction. Example: 20 seconds means that the swimmer will rest for 20 seconds after stopping the current instructions.</xss:documentation> </xss:annotation> </xss:element></pre>

Element restType / sinceStart

Namespace	https://github.com/bartneck/swiML
Annotations	The interval on which swimming instructions start. Example: on 1:30 means that the next instructions starts after 1:30 from starting the current instruction.
Diagram	
Type	xs:duration
Properties	content: simple
Source	<pre><xss:element name="sinceStart" type="xs:duration"> <xss:annotation> <xss:documentation>The interval on which swimming instructions start. Example: on 1:30 means that the next instructions starts after 1:30 from starting the current instruction.</xss:documentation> </xss:annotation> </xss:element></pre>

Element restType / sinceLastRest

Namespace	https://github.com/bartneck/swiML
Annotations	The time since the end of the last rest. This is useful when several instructions without a rest period are swum, followed by a since start type rest.

Diagram	A UML class diagram fragment. A box labeled "sinceLastRest" has a dependency arrow pointing to a box labeled "xs:duration". A callout box below "sinceLastRest" states: "The time since the end of the last rest. This is useful when several instructions without a rest period are swum,...". A callout box next to "xs:duration" states: "Built-in primitive type. The duration datatype represents a duration of time.".
Type	xs:duration
Properties	content: simple
Source	<pre><xs:element name="sinceLastRest" type="xs:duration"> <xs:annotation> <xs:documentation>The time since the end of the last rest. This is useful when several instructions without a rest period are swum, followed by a since start type rest.</xs:documentation> </xs:annotation> </xs:element></pre>

Element restType / inout

Namespace	https://github.com/bartneck/swiML
Annotations	Number of swimmers arriving. Example: 3rd in: Once the 3rd swimmer in the lane arrives, the 1st swimmer starts.
Diagram	A UML class diagram fragment. A box labeled "inOut" has a dependency arrow pointing to a box labeled "xs:nonNegativeInteger". A callout box below "inOut" states: "Number of swimmers arriving. Example: 3rd in: Once the 3rd swimmer in the lane arrives, the 1st swimmer starts.". A callout box next to "xs:nonNegativeInteger" states: "Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to...".
Type	xs:nonNegativeInteger
Properties	content: simple
Source	<pre><xs:element name="inOut" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>Number of swimmers arriving. Example: 3rd in: Once the 3rd swimmer in the lane arrives, the 1st swimmer starts.</xs:documentation> </xs:annotation> </xs:element></pre>

Element instructionGroup / intensity

Namespace	https://github.com/bartneck/swiML
Annotations	The intensity at which to swim the instruction.
Diagram	A UML class diagram fragment. A box labeled "intensity" has a dependency arrow pointing to a box labeled "intensityProfile". A callout box below "intensity" states: "The intensity at which to swim the instruction.". The "intensityProfile" box is marked as "Mixed" and "true". It contains two child boxes: "startIntensity" (Type: intensityType) and "stopIntensity" (Type: intensityType). A callout box below "intensityProfile" states: "The intensity of the instruction. When given at the lowest level just start intensity indicates a constant intensity if...".
Type	intensityProfile
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p> <p>mixed: true</p>
Model	startIntensity , stopIntensity{0,1}
Children	startIntensity, stopIntensity
Instance	<pre><intensity xmlns="https://github.com/bartneck/swiML"> <startIntensity>{1,1}</startIntensity> <stopIntensity>{0,1}</stopIntensity></pre>

	</intensity>
Source	<pre><xss:element name="intensity" minOccurs="0" maxOccurs="1" type="intensityProfile"> <xss:annotation> <xss:documentation>The intensity at which to swim the instruction.</xss:documentation> </xss:annotation> </xss:element></pre>

Element intensityProfile / startIntensity

Namespace	https://github.com/bartneck/swiML						
Diagram	<pre> classDiagram class startIntensity { <<Type intensityType>> } class intensityType { <<The intensity of the instructions.>> <<Effort in percentage. Example: 100 means maximum effort.>> <<Effort in training zone.>> <<Heart rate in percentage of maximum heart rate.>> } startIntensity < -- intensityType intensityType < -- percentageEffort intensityType < -- zone intensityType < -- percentageHeartRate </pre>						
Type	intensityType						
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	1	maxOccurs:	1
content:	complex						
minOccurs:	1						
maxOccurs:	1						
Model	percentageEffort zone percentageHeartRate						
Children	percentageEffort, percentageHeartRate, zone						
Instance	<pre> <startIntensity xmlns="https://github.com/bartneck/swiML"> <percentageEffort>{1,1}</percentageEffort> <zone>{1,1}</zone> <percentageHeartRate>{1,1}</percentageHeartRate> </startIntensity> </pre>						
Source	<pre><xss:element name="startIntensity" minOccurs="1" maxOccurs="1" type="intensityType"/></pre>						

Element intensityType / percentageEffort

Namespace	https://github.com/bartneck/swiML				
Annotations	Effort in percentage. Example: 100 means maximum effort.				
Diagram	<pre> classDiagram class percentageEffort { <<Effort in percentage. Example: 100 means maximum effort.>> } class percentType { <<The percent type specifies a value from 0 to 100.>> } percentageEffort < -- percentType </pre>				
Type	percentType				
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> </table>	content:	simple		
content:	simple				
Facets	<table border="1"> <tr> <td>maxInclusive</td> <td>100</td> </tr> <tr> <td>minInclusive</td> <td>0</td> </tr> </table>	maxInclusive	100	minInclusive	0
maxInclusive	100				
minInclusive	0				
Source	<pre> <xss:element name="percentageEffort" type="percentType"> <xss:annotation> <xss:documentation>Effort in percentage. Example: 100 means maximum effort.</xss:documentation> </xss:annotation> </xss:element> </pre>				

Element intensityType / zone

Namespace	https://github.com/bartneck/swiML
-----------	-----------------------------------

Annotations	Effort in training zone.										
Diagram	<pre> classDiagram class zone { Type zoneType } class zoneType { <<The intensity zone.>> } zone "1" --> "1" zoneType : zoneType "1" --> "1" zone : </pre> <p>The diagram shows a class named 'zone' with an attribute 'zoneType'. A self-referencing association from 'zone' to 'zoneType' is labeled with a note: 'Effort in training zone.' Another association from 'zoneType' back to 'zone' is labeled with a note: 'The intensity zone.'</p>										
Type	zoneType										
Properties	content: simple										
Facets	<table border="1"> <tr><td>enumeration</td><td>easy</td></tr> <tr><td>enumeration</td><td>threshold</td></tr> <tr><td>enumeration</td><td>endurance</td></tr> <tr><td>enumeration</td><td>racePace</td></tr> <tr><td>enumeration</td><td>max</td></tr> </table>	enumeration	easy	enumeration	threshold	enumeration	endurance	enumeration	racePace	enumeration	max
enumeration	easy										
enumeration	threshold										
enumeration	endurance										
enumeration	racePace										
enumeration	max										
Source	<pre> <xss:element name="zone" type="zoneType"> <xss:annotation> <xss:documentation>Effort in training zone.</xss:documentation> </xss:annotation> </xss:element> </pre>										

Element intensityType / percentageHeartRate

Namespace	https://github.com/bartneck/swiML				
Annotations	Heart rate in percentage of maximum heart rate.				
Diagram	<pre> classDiagram class percentageHeartRate { Type percentType } class percentType { <<The percent type specifies a value from 0 to 100.>> } percentageHeartRate "1" --> "1" percentType : percentType "1" --> "1" percentageHeartRate : </pre> <p>The diagram shows a class named 'percentageHeartRate' with an attribute 'percentType'. A self-referencing association from 'percentageHeartRate' to 'percentType' is labeled with a note: 'Heart rate in percentage of maximum heart rate.' Another association from 'percentType' back to 'percentageHeartRate' is labeled with a note: 'The percent type specifies a value from 0 to 100.'</p>				
Type	percentType				
Properties	content: simple				
Facets	<table border="1"> <tr><td>maxInclusive</td><td>100</td></tr> <tr><td>minInclusive</td><td>0</td></tr> </table>	maxInclusive	100	minInclusive	0
maxInclusive	100				
minInclusive	0				
Source	<pre> <xss:element name="percentageHeartRate" type="percentType"> <xss:annotation> <xss:documentation>Heart rate in percentage of maximum heart rate.</xss:documentation> </xss:annotation> </xss:element> </pre>				

Element intensityProfile / stopIntensity

Namespace	https://github.com/bartneck/swiML
Diagram	<pre> classDiagram class intensityType { <<The intensity of the instructions.>> } class percentageEffort { Type percentType } class zone { Type zoneType } class stopIntensity { Type intensityType } stopIntensity "1" --> "1" intensityType : intensityType "1" --> "1" percentageEffort : percentageEffort "1" --> "1" zone : zone "1" --> "1" zoneType : </pre> <p>The diagram illustrates the inheritance path of the 'intensityType' element. It starts with 'stopIntensity' which has 'intensityType' as its type. 'intensityType' is associated with 'percentageEffort' (labeled 'The intensity of the instructions.') via a self-referencing association. 'percentageEffort' is associated with 'zone' via a self-referencing association. Finally, 'zone' is associated with 'zoneType' via a self-referencing association. Notes along the associations include: 'Effort in percentage. Example: 100 means maximum effort.', 'Effort in training zone.', and 'Heart rate in percentage of maximum heart rate.'</p>
Type	intensityType
Properties	content: complex

	minOccurs: 0 maxOccurs: 1
Model	percentageEffort zone percentageHeartRate
Children	percentageEffort, percentageHeartRate, zone
Instance	<pre><stopIntensity xmlns="https://github.com/bartneck/swiML"> <percentageEffort>{1,1}</percentageEffort> <zone>{1,1}</zone> <percentageHeartRate>{1,1}</percentageHeartRate> </stopIntensity></pre>
Source	<pre><xss:element name="stopIntensity" minOccurs="0" maxOccurs="1" type="intensityType"/></pre>

Element instructionGroup / breath

Namespace	https://github.com/bartneck/swiML
Annotations	Number of arm strokes per breath.
Diagram	<p>The diagram shows a class named 'breath' with a 'Type' association to 'xs:nonNegativeInteger'. A callout box indicates that 'Number of arm strokes per breath.' is a built-in derived type derived from 'integer' by setting the value of 'minInclusive' to...</p>
Type	xs:nonNegativeInteger
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><xss:element name="breath" minOccurs="0" maxOccurs="1" type="xs:nonNegativeInteger"> <xss:annotation> <xss:documentation>Number of arm strokes per breath.</xss:documentation> </xss:annotation> </xss:element></pre>

Element instructionGroup / underwater

Namespace	https://github.com/bartneck/swiML
Annotations	True if swimming under water.
Diagram	<p>The diagram shows a class named 'underwater' with a 'Type' association to 'xs:boolean'. A callout box indicates that 'True if swimming under water.' is a built-in primitive type defining the boolean values true and false.</p>
Type	xs:boolean
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><xss:element name="underwater" minOccurs="0" maxOccurs="1" type="xs:boolean"> <xss:annotation> <xss:documentation>True if swimming under water.</xss:documentation> </xss:annotation> </xss:element></pre>

Element instructionGroup / equipment

Namespace	https://github.com/bartneck/swiML
Annotations	Equipment to be used, such as board or pads.
Diagram	<p>The diagram shows a class named 'equipment' with a 'Type' association to 'equipmentType'. A callout box indicates that 'Equipment to be used, such as board or pads.' is the type of equipment.</p>
Type	equipmentType

Properties	content:	simple
	minOccurs:	0
	maxOccurs:	unbounded
Facets	enumeration	board
	enumeration	pads
	enumeration	pullBuoy
	enumeration	fins
	enumeration	snorkle
	enumeration	chute
	enumeration	stretchCord
Source	<pre><xs:element name="equipment" minOccurs="0" maxOccurs="unbounded" type="equipmentType"> <xs:annotation> <xs:documentation>Equipment to be used, such as board or pads.</xs:documentation> </xs:annotation> </xs:element></pre>	

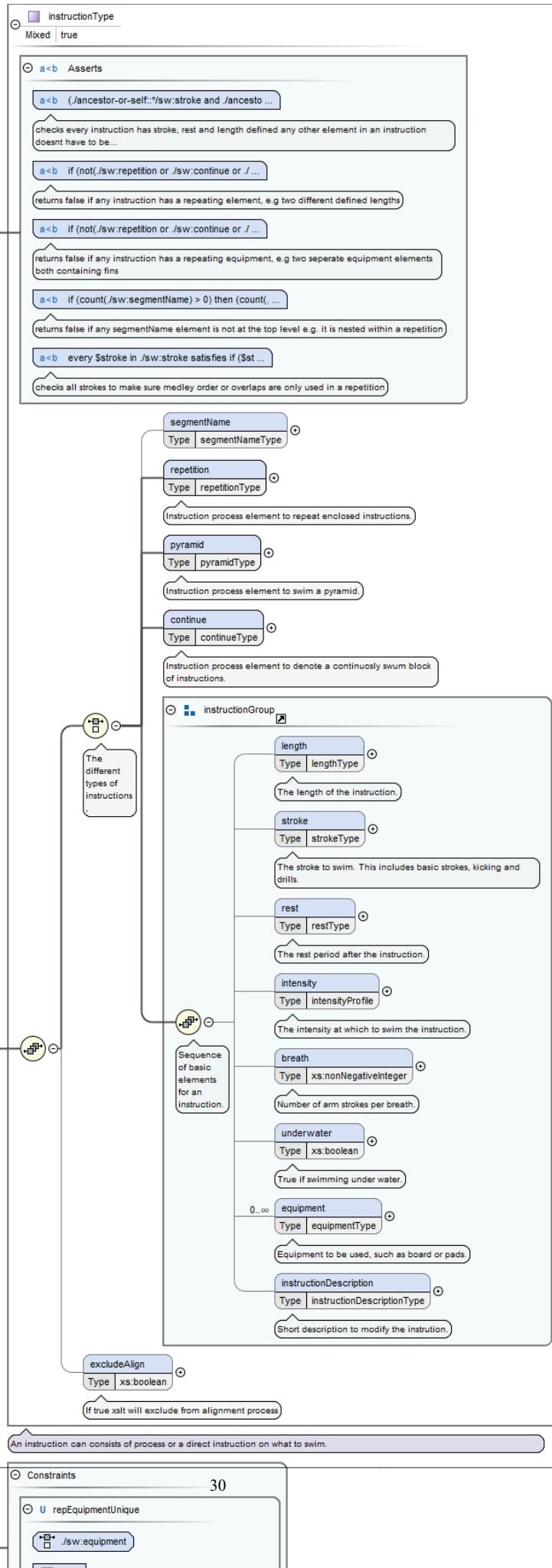
Element instructionGroup / instructionDescription

Namespace	https://github.com/bartneck/swiML							
Annotations	Short description to modify the instrution.							
Diagram	<p>The diagram shows a UML class structure. On the left, there is a class named "instructionDescription" with a compartment labeled "Type". An association line connects this class to another class on the right labeled "instructionDescriptionType". A note below the association line states "Short description to modify the instrution.". A separate note to the right of the association line states "The length of the description text is constraint in length."</p>							
Type	instructionDescriptionType							
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>		content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple							
minOccurs:	0							
maxOccurs:	1							
Facets	maxLength	100						
Source	<pre><xs:element name="instructionDescription" type="instructionDescriptionType" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>Short description to modify the instrution.</xs:documentation> </xs:annotation> </xs:element></pre>							

Element repetitionType / instruction

Namespace	https://github.com/bartneck/swiML	
Annotations	A repetition can include any number of instructions, which in turn can contain repetitions or continues.	

Diagram



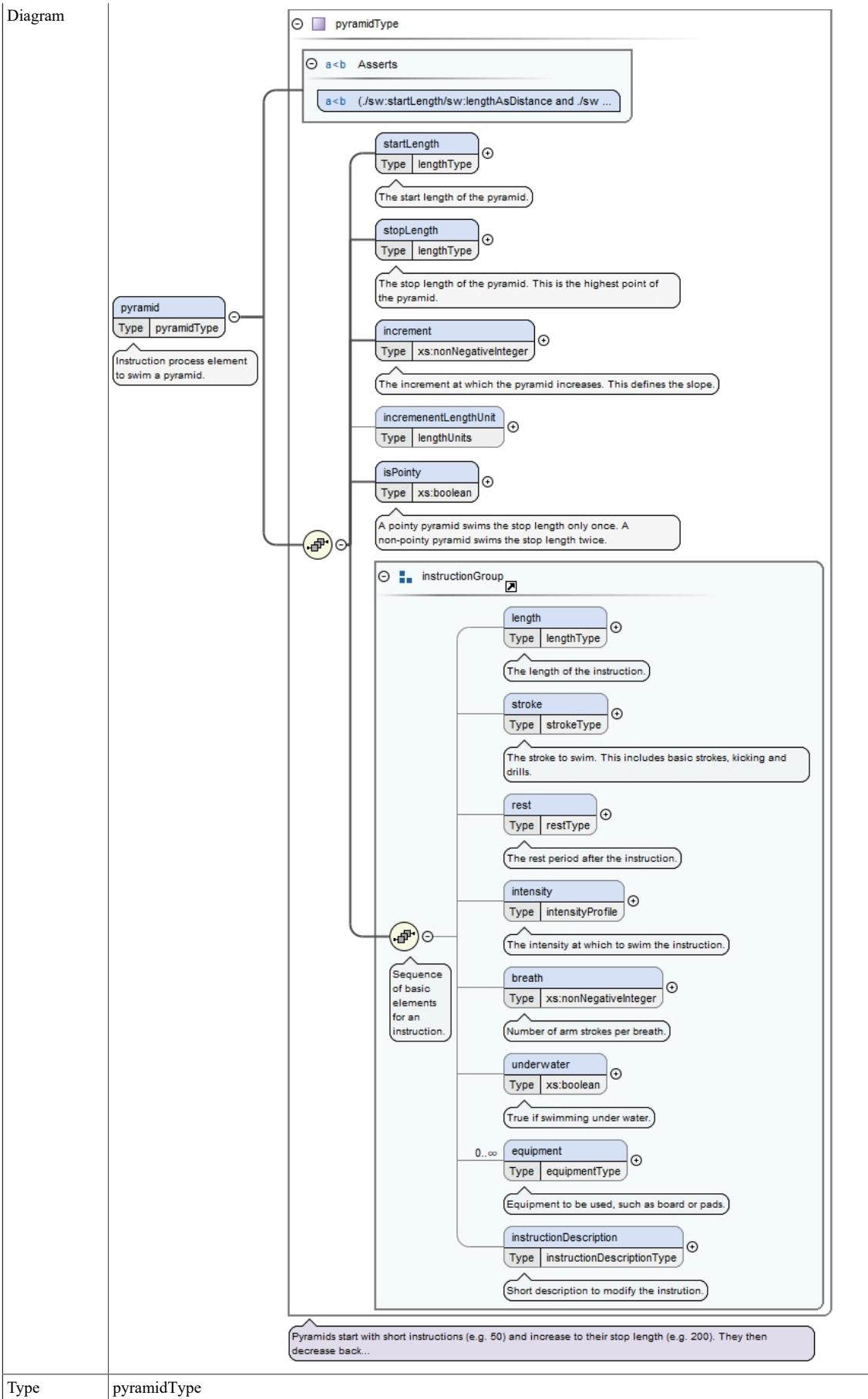
Type	instructionType	
Properties	content:	complex
	minOccurs:	1
	maxOccurs:	unbounded
	mixed:	true
Model	(segmentName{0,1} repetition pyramid continue (length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1})) , excludeAlign{0,1}	
Children	breath, continue, equipment, excludeAlign, instructionDescription, intensity, length, pyramid, repetition, rest, segmentName, stroke, underwater	
Instance	<pre><instruction xmlns="https://github.com/bartneck/swiML"> <segmentName>{0,1}</segmentName> <repetition>{1,1}</repetition> <pyramid>{1,1}</pyramid> <continue>{1,1}</continue> <length>{0,1}</length> <stroke>{0,1}</stroke> <rest>{0,1}</rest> <intensity>{0,1}</intensity> <breath>{0,1}</breath> <underwater>{0,1}</underwater> <equipment>{0,unbounded}</equipment> <instructionDescription>{0,1}</instructionDescription> <excludeAlign>{0,1}</excludeAlign> </instruction></pre>	
Asserts	<p>Test</p> <pre>(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName</pre> <p>checks every instruction has stroke, rest and length defined any other element in an instruction doesnt have to be defined</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in ./* satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'length' or name() = 'stroke' or name() = 'rest' or name() = 'intensity' or name() = 'breath' or name() = 'underwater'] satisfies not(name(\$element) = name(\$match))) else (true())</pre> <p>returns false if any instruction has a repeating element, e.g two different defined lengths</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in ./*[name() = 'equipment'] satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'equipment'] satisfies not(\$element/text() = \$match/text())) else (true())</pre> <p>returns false if any instruction has a repeating equipment, e.g two seperate equipment elements both containing fins</p> <pre>if(count(.//sw:segmentName) > 0) then (count(.//sw:segmentName/../../ancestor::*)) = 0 else (true())</pre> <p>returns false if any segmentName element is not at the top level e.g. it is nested within a repetition</p> <pre>every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOverlap' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'reverseIndividualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOverlap' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'reverseIndividualMedleyOrder') then (\$stroke/ancestor::*//sw:repetition) or (\$stroke/ancestor::*//sw:continue/sw:continueLength) else (\$stroke/parent::*))</pre>	<p>XPath default namespace</p>

	Test	XPath default namespace			
	checks all strokes to make sure medley order or overlaps are only used in a repetition				
Identity constraints	QName	Type	Refer	Selector	Field(s)
	repEquipmentUnique	unique		./sw:equipment	.
Source	<pre><xs:element name="instruction" minOccurs="1" maxOccurs="unbounded" type="instructionType"> <xs:annotation> <xs:documentation>A repetition can include any number of instructions, which in turn can contain repetitions or continues.</xs:documentation> </xs:annotation> <xs:unique name="repEquipmentUnique"> <xs:annotation> <xs:documentation>Ensures all equipment values in an instruction are unique</xs:documentation> </xs:annotation> <xs:selector xpath=".//sw:equipment" /> <xs:field xpath=".//." /> </xs:unique> </xs:element></pre>				

Element instructionType / pyramid

Namespace	https://github.com/bartneck/swiML
Annotations	Instruction process element to swim a pyramid.

Diagram



Type	pyramidType
------	-------------

Properties	content: complex	
Model	startLength , stopLength , increment , incremenentLengthUnit{0,1} , isPointy , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1}	
Children	breath, equipment, incremenentLengthUnit, increment, instructionDescription, intensity, isPointy, length, rest, startLength, sto-pLength, stroke, underwater	
Instance	<pre><pyramid xmlns="https://github.com/bartneck/swiML"> <startLength>{1,1}</startLength> <stopLength>{1,1}</stopLength> <increment>{1,1}</increment> <incremenentLengthUnit>{0,1}</incremenentLengthUnit> <isPointy>{1,1}</isPointy> <length>{0,1}</length> <stroke>{0,1}</stroke> <rest>{0,1}</rest> <intensity>{0,1}</intensity> <breath>{0,1}</breath> <underwater>{0,1}</underwater> <equipment>{0,unbounded}</equipment> <instructionDescription>{0,1}</instructionDescription> </pyramid></pre>	
Asserts	Test <code>(./sw:startLength/sw:lengthAsDistance and ./sw:sto-pLength/sw:lengthAsDistance) or (./sw:startLength/sw:lengthAsLaps and ./sw:stopLength/sw:lengthAsLaps) or (./sw:s-tartLength/sw:lengthAsTime and ./sw:stopLength/sw:lengthAs-Time)</code>	XPath default namespace
Source	<pre><xss:element name="pyramid" type="pyramidType"> <xss:annotation> <xss:documentation>Instruction process element to swim a pyramid.</xss:documentation> </xss:annotation> </xss:element></pre>	

Element pyramidType / startLength

Namespace	https://github.com/bartneck/swiML								
Annotations	The start length of the pyramid.								
Diagram	<pre> classDiagram class startLength { <<start length of the pyramid.>> } class lengthType { <<length for a swimming instruction.>> } startLength "0..1" --> "1..1" lengthType lengthType "0..1" --> "1..1" lengthAsDistance { <<Length of instruction as distance.>> Type xs:nonNegativeInteger } lengthType "0..1" --> "1..1" lengthAsTime { <<Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30.>> Type xs:duration } lengthType "0..1" --> "1..1" lengthAsLaps { <<Length of instruction in number of laps.>> Type xs:nonNegativeInteger } </pre>								
Type	lengthType								
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>mixed:</td> <td>true</td> </tr> </table>	content:	complex	minOccurs:	1	maxOccurs:	1	mixed:	true
content:	complex								
minOccurs:	1								
maxOccurs:	1								
mixed:	true								
Model	lengthAsDistance lengthAsTime lengthAsLaps								
Children	lengthAsDistance, lengthAsLaps, lengthAsTime								
Instance	<pre><startLength xmlns="https://github.com/bartneck/swiML"> <lengthAsDistance>{1,1}</lengthAsDistance> <lengthAsTime>{1,1}</lengthAsTime> <lengthAsLaps>{1,1}</lengthAsLaps></pre>								

	</startLength>
Source	<pre><xs:element name="startLength" minOccurs="1" maxOccurs="1" type="lengthType"> <xs:annotation> <xs:documentation>The start length of the pyramid.</xs:documentation> </xs:annotation> </xs:element></pre>

Element pyramidType / stopLength

Namespace	https://github.com/bartneck/swiML								
Annotations	The stop length of the pyramid. This is the highest point of the pyramid.								
Diagram	<pre> classDiagram class stopLength { <<stopLength xmlns="https://github.com/bartneck/swiML">> <lengthAsDistance>{1,1}</lengthAsDistance> <lengthAsTime>{1,1}</lengthAsTime> <lengthAsLaps>{1,1}</lengthAsLaps> } stopLength < -- lengthType lengthType { <<lengthType>> Mixed true } lengthType < -- lengthAsDistance lengthType < -- lengthAsTime lengthType < -- lengthAsLaps lengthAsDistance { <<lengthAsDistance>> Type xs:nonNegativeInteger Length of instruction as distance. } lengthAsTime { <<lengthAsTime>> Type xs:duration Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30. } lengthAsLaps { <<lengthAsLaps>> Type xs:nonNegativeInteger Length of instruction in number of laps. } note over stopLength: The stop length of the pyramid. This is the highest point of the pyramid. note over lengthType: Length can be described as distance or time. </pre>								
Type	lengthType								
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>mixed:</td> <td>true</td> </tr> </table>	content:	complex	minOccurs:	1	maxOccurs:	1	mixed:	true
content:	complex								
minOccurs:	1								
maxOccurs:	1								
mixed:	true								
Model	lengthAsDistance lengthAsTime lengthAsLaps								
Children	lengthAsDistance, lengthAsLaps, lengthAsTime								
Instance	<pre><stopLength xmlns="https://github.com/bartneck/swiML"> <lengthAsDistance>{1,1}</lengthAsDistance> <lengthAsTime>{1,1}</lengthAsTime> <lengthAsLaps>{1,1}</lengthAsLaps> </stopLength></pre>								
Source	<pre><xs:element name="stopLength" minOccurs="1" maxOccurs="1" type="lengthType"> <xs:annotation> <xs:documentation>The stop length of the pyramid. This is the highest point of the pyramid.</xs:documentation> </xs:annotation> </xs:element></pre>								

Element pyramidType / increment

Namespace	https://github.com/bartneck/swiML						
Annotations	The increment at which the pyramid increases. This defines the slope.						
Diagram	<pre> class increment { <<increment>> Type xs:nonNegativeInteger The increment at which the pyramid increases. This defines the slope. } increment < -- xs:nonNegativeInteger note over increment: The increment at which the pyramid increases. This defines the slope. note over xs:nonNegativeInteger: Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to... </pre>						
Type	xs:nonNegativeInteger						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	1	maxOccurs:	1
content:	simple						
minOccurs:	1						
maxOccurs:	1						

Source	<pre><xs:element name="increment" minOccurs="1" maxOccurs="1" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The increment at which the pyramid increases. This defines the slope.</xs:documentation> </xs:annotation> </xs:element></pre>
--------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Element pyramidType / incrementLengthUnit

Namespace	https://github.com/bartneck/swiML								
Diagram									
Type	lengthUnits								
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1		
content:	simple								
minOccurs:	0								
maxOccurs:	1								
Facets	<table border="1"> <tr> <td>enumeration</td> <td>meters</td> </tr> <tr> <td>enumeration</td> <td>kilometers</td> </tr> <tr> <td>enumeration</td> <td>miles</td> </tr> <tr> <td>enumeration</td> <td>yards</td> </tr> </table>	enumeration	meters	enumeration	kilometers	enumeration	miles	enumeration	yards
enumeration	meters								
enumeration	kilometers								
enumeration	miles								
enumeration	yards								
Source	<pre><xs:element name="incrementLengthUnit" type="lengthUnits" minOccurs="0" maxOccurs="1"/></pre>								

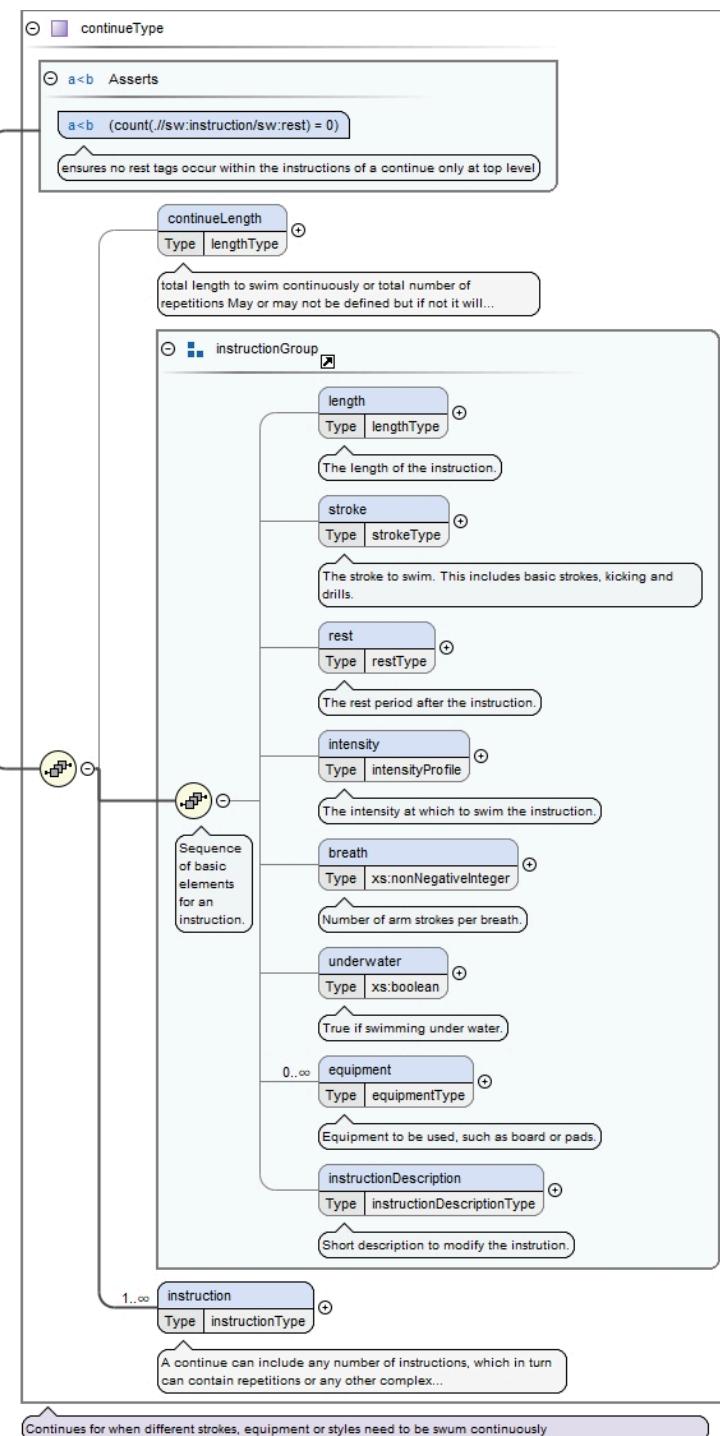
Element pyramidType / isPointy

Namespace	https://github.com/bartneck/swiML						
Annotations	A pointy pyramid swims the stop length only once. A non-pointy pyramid swims the stop length twice.						
Diagram							
Type	xs:boolean						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	1	maxOccurs:	1
content:	simple						
minOccurs:	1						
maxOccurs:	1						
Source	<pre><xs:element name="isPointy" minOccurs="1" maxOccurs="1" type="xs:boolean"> <xs:annotation> <xs:documentation>A pointy pyramid swims the stop length only once. A non-pointy pyramid swims the stop length twice.</xs:documentation> </xs:annotation> </xs:element></pre>						

Element instructionType / continue

Namespace	https://github.com/bartneck/swiML
Annotations	Instruction process element to denote a continuously swum block of instructions.

Diagram



Type	continueType
Properties	content: complex
Model	continueLength{0,1} , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1} , instruction+
Children	breath, continueLength, equipment, instruction, instructionDescription, intensity, length, rest, stroke, underwater
Instance	<pre><continue xmlns="https://github.com/bartneck/swiML"> <continueLength>{0,1}</continueLength> <length>{0,1}</length> <stroke>{0,1}</stroke> <rest>{0,1}</rest> <intensity>{0,1}</intensity> <breath>{0,1}</breath> <underwater>{0,1}</underwater> <equipment>{0,unbounded}</equipment></pre>

	<pre><instructionDescription>{0,1}</instructionDescription> <instruction>{1,unbounded}</instruction> </continue></pre>	
Asserts	Test <pre>(count(..//sw:instruction/sw:rest) = 0)</pre> <p>ensures no rest tags occur within the instructions of a continue only at top level</p>	XPath default namespace
Source	<pre><xs:element name="continue" type="continueType"> <xs:annotation> <xs:documentation>Instruction process element to denote a continuosly swum block of instructions.</xs:documentation> </xs:annotation> </xs:element></pre>	

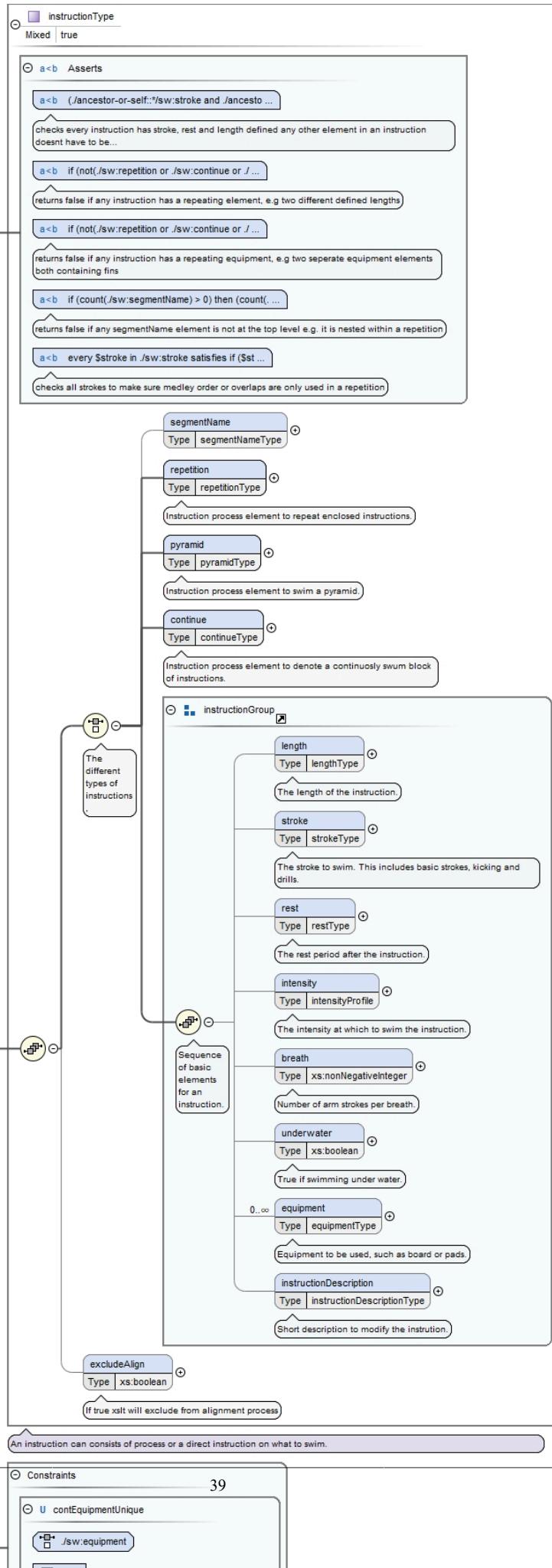
Element continueType / continueLength

Namespace	https://github.com/bartneck/swiML								
Annotations	total length to swim continuously or total number of repetitions May or may not be defined but if not it will automatically calculated from given instructions								
Diagram	<pre> classDiagram class continueLength { <<lengthType>> mixed true <<lengthAsDistance>> xs:nonNegativeInteger <<lengthAsTime>> xs:duration <<lengthAsLaps>> xs:nonNegativeInteger } </pre> <p>The diagram shows a UML class named <code>continueLength</code> which is a <code>Mixed</code> type (<code>lengthType</code>). It has three associations with other types: <code>lengthAsDistance</code> (type <code>xs:nonNegativeInteger</code>), <code>lengthAsTime</code> (type <code>xs:duration</code>), and <code>lengthAsLaps</code> (type <code>xs:nonNegativeInteger</code>). A note states: "Length can be described as distance or time." Another note specifies: "Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30." A summary note at the bottom says: "The length for a swimming instruction."</p>								
Type	lengthType								
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> <tr> <td>mixed:</td> <td>true</td> </tr> </table>	content:	complex	minOccurs:	0	maxOccurs:	1	mixed:	true
content:	complex								
minOccurs:	0								
maxOccurs:	1								
mixed:	true								
Model	lengthAsDistance lengthAsTime lengthAsLaps								
Children	lengthAsDistance, lengthAsLaps, lengthAsTime								
Instance	<pre><continueLength xmlns="https://github.com/bartneck/swiML"> <lengthAsDistance>{1,1}</lengthAsDistance> <lengthAsTime>{1,1}</lengthAsTime> <lengthAsLaps>{1,1}</lengthAsLaps> </continueLength></pre>								
Source	<pre><xs:element name="continueLength" minOccurs="0" maxOccurs="1" type="lengthType"> <xs:annotation> <xs:documentation>total length to swim continuously or total number of repetitions May or may not be defined but if not it will automatically calculated from given instructions</xs:documentation> </xs:annotation> </xs:element></pre>								

Element continueType / instruction

Namespace	https://github.com/bartneck/swiML
Annotations	A continue can include any number of instructions, which in turn can contain repetitions or any other complex instruction type.

Diagram



Type	instructionType	
Properties	content:	complex
	minOccurs:	1
	maxOccurs:	unbounded
	mixed:	true
Model	(segmentName{0,1} repetition pyramid continue (length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1})) , excludeAlign{0,1}	
Children	breath, continue, equipment, excludeAlign, instructionDescription, intensity, length, pyramid, repetition, rest, segmentName, stroke, underwater	
Instance	<pre><instruction xmlns="https://github.com/bartneck/swiML"> <segmentName>{0,1}</segmentName> <repetition>{1,1}</repetition> <pyramid>{1,1}</pyramid> <continue>{1,1}</continue> <length>{0,1}</length> <stroke>{0,1}</stroke> <rest>{0,1}</rest> <intensity>{0,1}</intensity> <breath>{0,1}</breath> <underwater>{0,1}</underwater> <equipment>{0,unbounded}</equipment> <instructionDescription>{0,1}</instructionDescription> <excludeAlign>{0,1}</excludeAlign> </instruction></pre>	
Asserts	<p>Test</p> <pre>(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName</pre> <p>checks every instruction has stroke, rest and length defined any other element in an instruction doesnt have to be defined</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in ./* satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'length' or name() = 'stroke' or name() = 'rest' or name() = 'intensity' or name() = 'breath' or name() = 'underwater'] satisfies not(name(\$element) = name(\$match))) else (true())</pre> <p>returns false if any instruction has a repeating element, e.g two different defined lengths</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in ./*[name() = 'equipment'] satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'equipment'] satisfies not(\$element/text() = \$match/text())) else (true())</pre> <p>returns false if any instruction has a repeating equipment, e.g two seperate equipment elements both containing fins</p> <pre>if(count(.//sw:segmentName) > 0) then (count(.//sw:segmentName/../../ancestor::*) = 0) else (true())</pre> <p>returns false if any segmentName element is not at the top level e.g. it is nested within a repetition</p> <pre>every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOverlap' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'reverseIndividualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOverlap' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'reverseIndividualMedleyOrder') then (\$stroke/ancestor::*/sw:repetition) or (\$stroke/ancestor::*/sw:continue/sw:continueLength) else (\$stroke/parent::*)</pre>	<p>XPath default namespace</p>

	Test	XPath default namespace			
	checks all strokes to make sure medley order or overlaps are only used in a repetition				
Identity constraints	QName	Type	Refer	Selector	Field(s)
	contEquipmentUnique	unique		./sw:equipment	.
Source	<pre><xs:element name="instruction" minOccurs="1" maxOccurs="unbounded" type="instructionType"> <xs:annotation> <xs:documentation>A continue can include any number of instructions, which in turn can contain repetitions or any other complex instruction type.</xs:documentation> </xs:annotation> <xs:unique name="contEquipmentUnique"> <xs:annotation> <xs:documentation>Ensures all equipment values in an instruction are unique</xs:documentation> </xs:annotation> <xs:selector xpath=".//sw:equipment" /> <xs:field xpath=".//." /> </xs:unique> </xs:element></pre>				

Element instructionType / excludeAlign

Namespace	https://github.com/bartneck/swiML						
Annotations	If true xsslt will exclude from alignment process						
Diagram	<p>The diagram shows a UML class named 'excludeAlign' with a 'Type' association to 'xs:boolean'. A callout box for 'excludeAlign' states: 'If true xsslt will exclude from alignment process'. A callout box for 'xs:boolean' states: 'Built-in primitive type. It defines the boolean values true and false.'</p>						
Type	xs:boolean						
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> <tr> <td>maxOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	1
content:	simple						
minOccurs:	0						
maxOccurs:	1						
Source	<pre><xs:element name="excludeAlign" type="xs:boolean" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>If true xsslt will exclude from alignment process</xs:documentation> </xs:annotation> </xs:element></pre>						

Simple Type(s)

Simple Type titleString

Namespace	https://github.com/bartneck/swiML
Annotations	The length of the title is constraint in length.
Diagram	<p>The diagram shows a UML class named 'titleString' with a 'restriction of xs:string' association to 'xs:string'. A callout box for 'titleString' states: 'The length of the title is constraint in length.'. A callout box for 'xs:string' states: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>
Type	restriction of xs:string
Facets	maxLength 60
Used by	Element program/title
Source	<pre><xs:simpleType name="titleString"> <xs:annotation> <xs:documentation>The length of the title is constraint in length.</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:maxLength value="60" /> </xs:restriction> </xs:simpleType></pre>

Simple Type emailAddress

Namespace	https://github.com/bartneck/swiML
-----------	-----------------------------------

Annotations	The pattern checks for valid email addresses.
Diagram	
Type	restriction of xs:string
Facets	pattern $[^@]+@[^\ .]+\.\.+$
Used by	Element program/author/email
Source	<pre><xs:simpleType name="emailAddress"> <xs:annotation> <xs:documentation>The pattern checks for valid email addresses.</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:pattern value="[^@]+@[^\ .]+\.\.+"/> </xs:restriction> </xs:simpleType></pre>

Simple Type descriptionString

Namespace	https://github.com/bartneck/swiML
Annotations	The length of the description text is constraint in length.
Diagram	
Type	restriction of xs:string
Facets	maxLength 400
Used by	Element program/programDescription
Source	<pre><xs:simpleType name="descriptionString"> <xs:annotation> <xs:documentation>The length of the description text is constraint in length.</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:maxLength value="400"/> </xs:restriction> </xs:simpleType></pre>

Simple Type lengthUnits

Namespace	https://github.com/bartneck/swiML								
Annotations	The unit of measurement for the length of the target pool (meter or yards).								
Diagram									
Type	restriction of xs:string								
Facets	<table border="1"> <tr> <td>enumeration</td> <td>meters</td> </tr> <tr> <td>enumeration</td> <td>kilometers</td> </tr> <tr> <td>enumeration</td> <td>miles</td> </tr> <tr> <td>enumeration</td> <td>yards</td> </tr> </table>	enumeration	meters	enumeration	kilometers	enumeration	miles	enumeration	yards
enumeration	meters								
enumeration	kilometers								
enumeration	miles								
enumeration	yards								
Used by	Elements program/lengthUnit, pyramidType/incremenentLengthUnit								
Source	<pre><xs:simpleType name="lengthUnits"> <xs:annotation> <xs:documentation>The unit of measurement for the length of the target pool (meter or yards).</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:enumeration value="meters"/> <xs:enumeration value="kilometers"/> </xs:restriction> </xs:simpleType></pre>								

```

<xs:enumeration value="miles" />
<xs:enumeration value="yards" />
</xs:restriction>
</xs:simpleType>

```

Simple Type numeralSystems

Namespace	https://github.com/bartneck/swiML					
Annotations	Numeral system type to be used for displaying the document					
Diagram	<p>Diagram illustrating the relationship between the simple type <code>numeralSystems</code> and the built-in primitive type <code>xs:string</code>. The <code>numeralSystems</code> type is shown as a rounded rectangle with a hollow center, connected by a line with an open circle to the <code>xs:string</code> type, also in a rounded rectangle with a hollow center. A callout box points to the <code>numeralSystems</code> type with the text: "Numeral system type to be used for displaying the document". Another callout box points to the <code>xs:string</code> type with the text: "Built-in primitive type. The string datatype represents character strings in XML."</p>					
Type	restriction of <code>xs:string</code>					
Facets	<table> <tr> <td>enumeration</td> <td>decimal</td> </tr> <tr> <td>enumeration</td> <td>roman</td> </tr> </table>		enumeration	decimal	enumeration	roman
enumeration	decimal					
enumeration	roman					
Used by	Element <code>program/numeralSystem</code>					
Source	<pre> <xs:simpleType name="numeralSystems"> <xs:annotation> <xs:documentation>Numeral system type to be used for displaying the document</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:enumeration value="decimal"/> <xs:enumeration value="roman"/> </xs:restriction> </xs:simpleType> </pre>					

Simple Type segmentNameType

Namespace	https://github.com/bartneck/swiML	
Diagram	<p>Diagram illustrating the relationship between the simple type <code>segmentNameType</code> and the built-in primitive type <code>xs:string</code>. The <code>segmentNameType</code> type is shown as a rounded rectangle with a hollow center, connected by a line with an open circle to the <code>xs:string</code> type, also in a rounded rectangle with a hollow center. A callout box points to the <code>xs:string</code> type with the text: "Built-in primitive type. The string datatype represents character strings in XML."</p>	
Type	restriction of <code>xs:string</code>	
Facets	maxLength 50	
Used by	Element <code>instructionType/segmentName</code>	
Source	<pre> <xs:simpleType name="segmentNameType"> <xs:restriction base="xs:string"> <xs:maxLength value=" 50 "/> </xs:restriction> </xs:simpleType> </pre>	

Simple Type instructionDescriptionType

Namespace	https://github.com/bartneck/swiML	
Annotations	The length of the description text is constraint in length.	
Diagram	<p>Diagram illustrating the relationship between the simple type <code>instructionDescriptionType</code> and the built-in primitive type <code>xs:string</code>. The <code>instructionDescriptionType</code> type is shown as a rounded rectangle with a hollow center, connected by a line with an open circle to the <code>xs:string</code> type, also in a rounded rectangle with a hollow center. A callout box points to the <code>instructionDescriptionType</code> type with the text: "The length of the description text is constraint in length.". Another callout box points to the <code>xs:string</code> type with the text: "Built-in primitive type. The string datatype represents character strings in XML."</p>	
Type	restriction of <code>xs:string</code>	
Facets	maxLength 100	
Used by	Elements <code>instructionGroup/instructionDescription</code> , <code>repetitionType/repetitionDescription</code>	
Source	<pre> <xs:simpleType name="instructionDescriptionType"> <xs:annotation> <xs:documentation>The length of the description text is constraint in length.</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:maxLength value="100"/> </xs:restriction> </xs:simpleType> </pre>	

```

    </xs:restriction>
</xs:simpleType>

```

Simple Type standardStrokeType

Namespace	https://github.com/bartneck/swiML																																					
Diagram																																						
Type	restriction of xs:string																																					
Facets	<table border="1"> <tr><td>enumeration</td><td>butterfly</td></tr> <tr><td>enumeration</td><td>backstroke</td></tr> <tr><td>enumeration</td><td>breaststroke</td></tr> <tr><td>enumeration</td><td>freestyle</td></tr> <tr><td>enumeration</td><td>individualMedley</td></tr> <tr><td>enumeration</td><td>reverseIndividualMedley</td></tr> <tr><td>enumeration</td><td>individualMedleyOverlap</td></tr> <tr><td>enumeration</td><td>individualMedleyOrder</td></tr> <tr><td>enumeration</td><td>reverseIndividualMedley-Order</td></tr> <tr><td>enumeration</td><td>any</td></tr> <tr><td>enumeration</td><td>nr1</td></tr> <tr><td>enumeration</td><td>nr2</td></tr> <tr><td>enumeration</td><td>nr3</td></tr> <tr><td>enumeration</td><td>nr4</td></tr> <tr><td>enumeration</td><td>notButterfly</td></tr> <tr><td>enumeration</td><td>notBackstroke</td></tr> <tr><td>enumeration</td><td>notBreaststroke</td></tr> <tr><td>enumeration</td><td>notFreestyle</td></tr> </table>		enumeration	butterfly	enumeration	backstroke	enumeration	breaststroke	enumeration	freestyle	enumeration	individualMedley	enumeration	reverseIndividualMedley	enumeration	individualMedleyOverlap	enumeration	individualMedleyOrder	enumeration	reverseIndividualMedley-Order	enumeration	any	enumeration	nr1	enumeration	nr2	enumeration	nr3	enumeration	nr4	enumeration	notButterfly	enumeration	notBackstroke	enumeration	notBreaststroke	enumeration	notFreestyle
enumeration	butterfly																																					
enumeration	backstroke																																					
enumeration	breaststroke																																					
enumeration	freestyle																																					
enumeration	individualMedley																																					
enumeration	reverseIndividualMedley																																					
enumeration	individualMedleyOverlap																																					
enumeration	individualMedleyOrder																																					
enumeration	reverseIndividualMedley-Order																																					
enumeration	any																																					
enumeration	nr1																																					
enumeration	nr2																																					
enumeration	nr3																																					
enumeration	nr4																																					
enumeration	notButterfly																																					
enumeration	notBackstroke																																					
enumeration	notBreaststroke																																					
enumeration	notFreestyle																																					
Used by	Elements	drillType/drillStroke, kickStyle/standardKick, strokeType/standardStroke																																				
Source	<pre> <xs:simpleType name="standardStrokeType"> <xs:restriction base="xs:string"> <xs:enumeration value="butterfly"/> <xs:enumeration value="backstroke"/> <xs:enumeration value="breaststroke"/> <xs:enumeration value="freestyle"/> <xs:enumeration value="individualMedley"/> <xs:enumeration value="reverseIndividualMedley"/> <xs:enumeration value="individualMedleyOverlap"/> <xs:enumeration value="individualMedleyOrder"/> <xs:enumeration value="reverseIndividualMedleyOrder"/> <xs:enumeration value="any"/> <xs:enumeration value="nr1"/> <xs:enumeration value="nr2"/> <xs:enumeration value="nr3"/> <xs:enumeration value="nr4"/> <xs:enumeration value="notButterfly"/> <xs:enumeration value="notBackstroke"/> <xs:enumeration value="notBreaststroke"/> <xs:enumeration value="notFreestyle"/> </xs:restriction> </xs:simpleType> </pre>																																					

Simple Type orientationType

Namespace	https://github.com/bartneck/swiML	
Diagram		

Type	restriction of xs:string	
Facets	enumeration	front
	enumeration	back
	enumeration	left
	enumeration	right
	enumeration	side
	enumeration	vertical
	enumeration	waka
Used by	Element	kickStyle/orientation
Source	<pre><xs:simpleType name="orientationType"> <xs:restriction base="xs:string"> <xs:enumeration value="front"/> <xs:enumeration value="back"/> <xs:enumeration value="left"/> <xs:enumeration value="right"/> <xs:enumeration value="side"/> <xs:enumeration value="vertical"/> <xs:enumeration value="waka"/> </xs:restriction> </xs:simpleType></pre>	

Simple Type legMovementType

Namespace	https://github.com/bartneck/swiML	
Diagram	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>	
Type	restriction of xs:string	
Facets	enumeration	flutter
	enumeration	dolphin
	enumeration	scissor
Used by	Element	kickStyle/legMovement
Source	<pre><xs:simpleType name="legMovementType"> <xs:restriction base="xs:string"> <xs:enumeration value="flutter"/> <xs:enumeration value="dolphin"/> <xs:enumeration value="scissor"/> </xs:restriction> </xs:simpleType></pre>	

Simple Type drillNameType

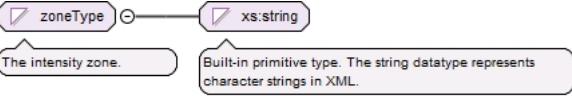
Namespace	https://github.com/bartneck/swiML	
Annotations	Drill names.	
Diagram	<p>Drill names.</p> <p>Built-in primitive type. The string datatype represents character strings in XML.</p>	
Type	restriction of xs:string	
Facets	enumeration	6KickDrill
	enumeration	8KickDrill
	enumeration	10KickDrill
	enumeration	12KickDrill
	enumeration	fingerTrails
	enumeration	123
	enumeration	bigDog
	enumeration	scull

	enumeration	singleArm
	enumeration	any
	enumeration	technic
	enumeration	dogPaddle
	enumeration	tarzan
	enumeration	fist
	enumeration	2Kick1Pull
	enumeration	3Kick1Pull
	enumeration	2Pull1Kick
	enumeration	3Pull1Kick
	enumeration	3Right3Left
	enumeration	2Right2Left
	enumeration	other
Used by	Element	drillType/drillName
Source	<pre><xs:simpleType name="drillNameType"> <xs:annotation> <xs:documentation>Drill names.</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:enumeration value="6KickDrill"/> <xs:enumeration value="8KickDrill"/> <xs:enumeration value="10KickDrill"/> <xs:enumeration value="12KickDrill"/> <xs:enumeration value="fingerTrails"/> <xs:enumeration value="123"/> <xs:enumeration value="bigDog"/> <xs:enumeration value="scull"/> <xs:enumeration value="singleArm"/> <xs:enumeration value="any"/> <xs:enumeration value="technic"/> <xs:enumeration value="dogPaddle"/> <xs:enumeration value="tarzan"/> <xs:enumeration value="fist"/> <xs:enumeration value="2Kick1Pull"/> <xs:enumeration value="3Kick1Pull"/> <xs:enumeration value="2Pull1Kick"/> <xs:enumeration value="3Pull1Kick"/> <xs:enumeration value="3Right3Left"/> <xs:enumeration value="2Right2Left"/> <xs:enumeration value="other"/> </xs:restriction> </xs:simpleType></pre>	

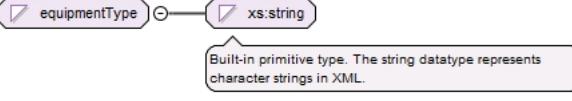
Simple Type `percentType`

Namespace	https://github.com/bartneck/swiML					
Annotations	The percent type specifies a value from 0 to 100.					
Diagram	<p>The percent type specifies a value from 0 to 100.</p> <p>Built-in primitive type. The decimal datatype represents arbitrary precision decimal numbers.</p>					
Type	restriction of xs:decimal					
Facets	<table border="1"> <tr> <td>maxInclusive</td> <td>100</td> </tr> <tr> <td>minInclusive</td> <td>0</td> </tr> </table>		maxInclusive	100	minInclusive	0
maxInclusive	100					
minInclusive	0					
Used by	Elements intensityType/percentageEffort, intensityType/percentageHeartRate					
Source	<pre><xs:simpleType name="percentType"> <xs:annotation> <xs:documentation>The percent type specifies a value from 0 to 100.</xs:documentation> </xs:annotation> <xs:restriction base="xs:decimal"> <xs:minInclusive value="0"/> <xs:maxInclusive value="100"/> </xs:restriction> </xs:simpleType></pre>					

Simple Type zoneType

Namespace	https://github.com/bartneck/swiML											
Annotations	The intensity zone.											
Diagram	 <p>The diagram shows a UML class named 'zoneType' represented by a rounded rectangle with a purple border. An association line connects it to another rounded rectangle with a purple border labeled 'xs:string'. A generalization symbol (a hollow circle with a line) is placed between them. A callout box points to 'zoneType' with the text 'The intensity zone.' and another callout box points to 'xs:string' with the text 'Built-in primitive type. The string datatype represents character strings in XML.'</p>											
Type	restriction of xs:string											
Facets	<table> <tr> <td>enumeration</td> <td>easy</td> </tr> <tr> <td>enumeration</td> <td>threshold</td> </tr> <tr> <td>enumeration</td> <td>endurance</td> </tr> <tr> <td>enumeration</td> <td>racePace</td> </tr> <tr> <td>enumeration</td> <td>max</td> </tr> </table>		enumeration	easy	enumeration	threshold	enumeration	endurance	enumeration	racePace	enumeration	max
enumeration	easy											
enumeration	threshold											
enumeration	endurance											
enumeration	racePace											
enumeration	max											
Used by	Element	intensityType/zone										
Source	<pre><xs:simpleType name="zoneType"> <xs:annotation> <xs:documentation>The intensity zone.</xs:documentation> </xs:annotation> <xs:restriction base="xs:string"> <xs:enumeration value="easy"/> <xs:enumeration value="threshold"/> <xs:enumeration value="endurance"/> <xs:enumeration value="racePace"/> <xs:enumeration value="max"/> </xs:restriction> </xs:simpleType></pre>											

Simple Type equipmentType

Namespace	https://github.com/bartneck/swiML															
Diagram	 <p>The diagram shows a UML class named 'equipmentType' represented by a rounded rectangle with a purple border. An association line connects it to another rounded rectangle with a purple border labeled 'xs:string'. A generalization symbol (a hollow circle with a line) is placed between them. A callout box points to 'equipmentType' with the text 'The intensity zone.' and another callout box points to 'xs:string' with the text 'Built-in primitive type. The string datatype represents character strings in XML.'</p>															
Type	restriction of xs:string															
Facets	<table> <tr> <td>enumeration</td> <td>board</td> </tr> <tr> <td>enumeration</td> <td>pads</td> </tr> <tr> <td>enumeration</td> <td>pullBuoy</td> </tr> <tr> <td>enumeration</td> <td>fins</td> </tr> <tr> <td>enumeration</td> <td>snorkle</td> </tr> <tr> <td>enumeration</td> <td>chute</td> </tr> <tr> <td>enumeration</td> <td>stretchCord</td> </tr> </table>		enumeration	board	enumeration	pads	enumeration	pullBuoy	enumeration	fins	enumeration	snorkle	enumeration	chute	enumeration	stretchCord
enumeration	board															
enumeration	pads															
enumeration	pullBuoy															
enumeration	fins															
enumeration	snorkle															
enumeration	chute															
enumeration	stretchCord															
Used by	Element	instructionGroup/equipment														
Source	<pre><xs:simpleType name="equipmentType"> <xs:restriction base="xs:string"> <xs:enumeration value="board"/> <xs:enumeration value="pads"/> <xs:enumeration value="pullBuoy"/> <xs:enumeration value="fins"/> <xs:enumeration value="snorkle"/> <xs:enumeration value="chute"/> <xs:enumeration value="stretchCord"/> </xs:restriction> </xs:simpleType></pre>															

Simple Type equipmentList

Namespace	https://github.com/bartneck/swiML	
Diagram	 <p>The diagram shows a UML class named 'equipmentList' represented by a rounded rectangle with a purple border. An association line connects it to another rounded rectangle with a purple border labeled 'equipmentType'. A composition symbol (a hollow circle with a line and a hollow diamond with a line) is placed between them.</p>	

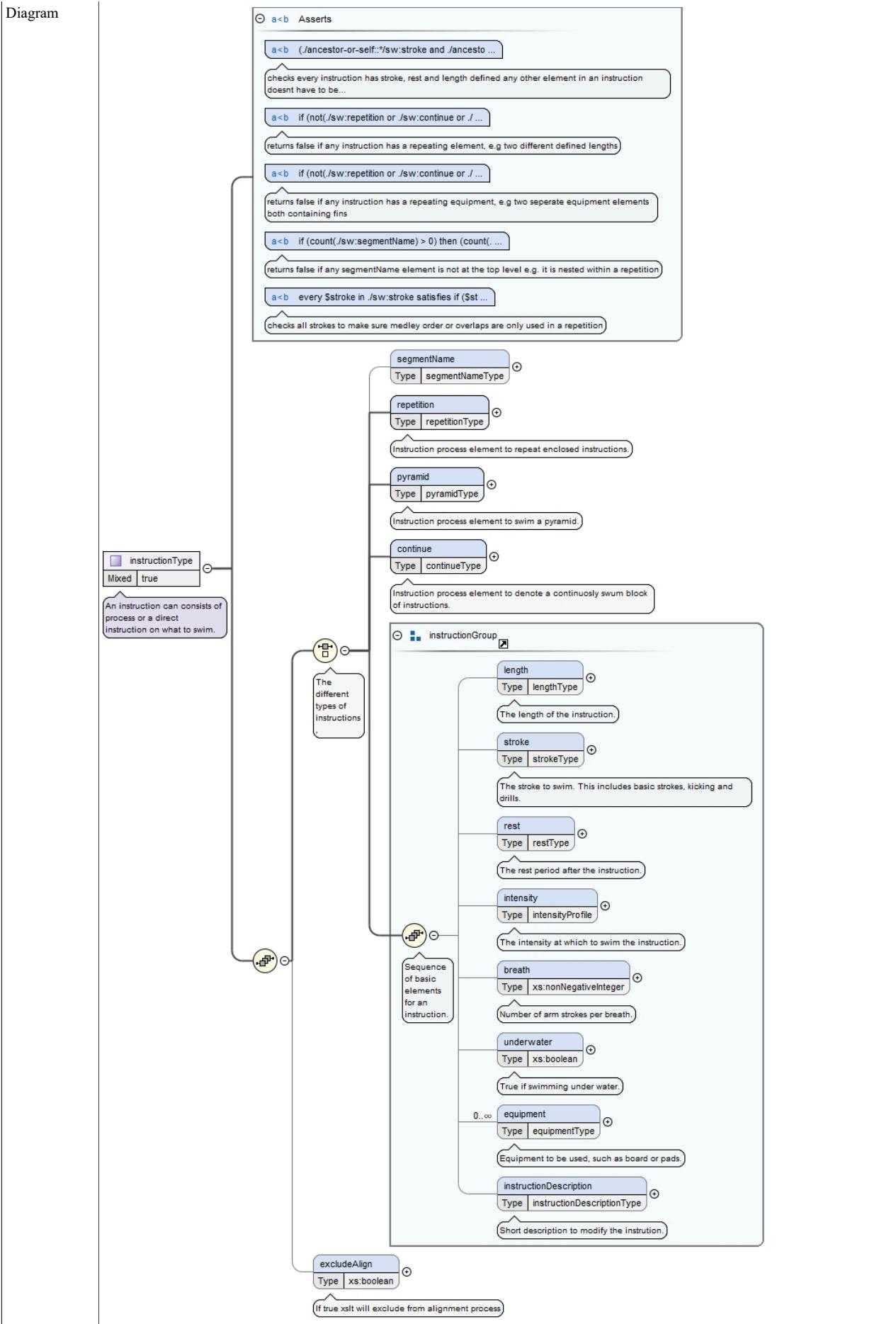
Type	list of equipmentType
Source	<pre><xs:simpleType name="equipmentList"> <xs:list itemType="equipmentType"/> </xs:simpleType></pre>

Complex Type(s)

Complex Type instructionType

Namespace	https://github.com/bartneck/swiML
Annotations	An instruction can consists of process or a direct instruction on what to swim.

Diagram



Properties	mixed: true	
Used by	Elements continueType/instruction, program/instruction, repetitionType/instruction	
Model	(segmentName{0,1} repetition pyramid continue (length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1})) , excludeAlign{0,1}	
Children	breath, continue, equipment, excludeAlign, instructionDescription, intensity, length, pyramid, repetition, rest, segmentName, stroke, underwater	
Asserts	<p>Test</p> <pre>(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName</pre> <p>checks every instruction has stroke, rest and length defined any other element in an instruction doesnt have to be defined</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in /* satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'length' or name() = 'stroke' or name() = 'rest' or name() = 'intensity' or name() = 'breath' or name() = 'underwater'] satisfies not(name(\$element) = name(\$match))) else (true())</pre> <p>returns false if any instruction has a repeating element, e.g two different defined lengths</p> <pre>if(not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)) then (every \$element in /*[name() = 'equipment'] satisfies (every \$match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'equipment'] satisfies not(\$element/text() = \$match/text())) else (true())</pre> <p>returns false if any instruction has a repeating equipment, e.g two seperate equipment elements both containing fins</p> <pre>if(count(.//sw:segmentName) > 0) then (count(.//sw:segmentName/../../ancestor::*) = 0) else (true())</pre> <p>returns false if any segmentName element is not at the top level e.g. it is nested within a repetition</p> <pre>every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOverlap' or \$stroke/sw:kicking/sw:standardKick = 'individualMedleyOrder' or \$stroke/sw:kicking/sw:standardKick = 'reverseIndividualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOverlap' or \$stroke/sw:drill/sw:drillStroke = 'individualMedleyOrder' or \$stroke/sw:drill/sw:drillStroke = 'reverseIndividualMedleyOrder') then (\$stroke/ancestor::*/sw:repetition) or (\$stroke/ancestor::*/sw:continue/sw:continueLength) else (\$stroke/parent::*)</pre> <p>checks all strokes to make sure medley order or overlaps are only used in a repetition</p>	<p>XPath default namespace</p>
Source	<pre><xss:complexType name="instructionType" mixed="true"> <xss:annotation> <xss:documentation>An instruction can consists of process or a direct instruction on what to swim.</xss:documentation> </xss:annotation> <xss:sequence> <xss:choice> <xss:annotation> <xss:documentation>The different types of instructions,</xss:documentation> </xss:annotation> <!-- ===== --> <!-- Process based elements for instructions --> <xss:element name="segmentName" minOccurs="0" maxOccurs="1" type="segmentNameType"/> <xss:element name="repetition" type="repetitionType"> <xss:annotation> <xss:documentation>Instruction process element to repeat enclosed instructions.</xss:documentation> </xss:annotation> </xss:element> </xss:choice> </xss:sequence> </xss:complexType></pre>	

```

</xs:annotation>
</xs:element>
<xs:element name="pyramid" type="pyramidType">
    <xs:documentation>Instruction process element to swim a pyramid.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="continue" type="continueType">
    <xs:annotation>
        <xs:documentation>Instruction process element to denote a continuosly swum block of
instructions.</xs:documentation>
    </xs:annotation>
</xs:element>
<!-- ===== -->
<!-- Direct instruction on what to swim -->
<xs:group ref="instructionGroup"/>
</xs:choice>
<xs:element name="excludeAlign" type="xs:boolean" minOccurs="0" maxOccurs="1">
    <xs:annotation>
        <xs:documentation>If true xslt will exclude from alignment process</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
<!-- checks every instruction has stroke, rest and length defined
any other element in an instruction doesnt have to be defined-->
<xs:assert test="
                (./ancestor-or-self::*/sw:stroke
                    or ./sw:repetition
                    or ./sw:segmentName)
            ">
    <xs:annotation>
        <xs:documentation>checks every instruction has stroke, rest and length defined any other
element in an instruction doesnt have to be defined</xs:documentation>
    </xs:annotation>
</xs:assert>
<!-- checks every instruction doesnt have repetitions of elements defined and cannot be extended
-->
<xs:assert test="
                if (not(./sw:repetition
                    or ./sw:pyramid
                    or ./sw:segmentName))
                    (
                        every $element in ./*
                            satisfies (
                                every $match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name()
= 'continue' or name() = 'pyramid'][not(./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'length' or name() = 'stroke' or name() = 'rest' or name() = 'intensity'
or name() = 'breath' or name() = 'underwater']
                                    satisfies
                                        not(name($element) = name($match))
                                )
                            else
                                (true())
                            )
                    )
            " or ./sw:continue
            then
            else
        <xs:annotation>
            <xs:documentation>returns false if any instruction has a repeating element, e.g two different
defined lengths</xs:documentation>
        </xs:annotation>
</xs:assert>
<xs:assert test="
                if (not(./sw:repetition
                    or ./sw:pyramid
                    or ./sw:segmentName))
                    (
                        every $element in ./*
                            satisfies (
                                every $match in ./ancestor::*[name() = 'instruction' or name()
= 'repetition' or name() = 'continue' or name() = 'pyramid'][not(./sw:repetition or ./sw:continue
or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'equipment']
                                    satisfies
                                        not($element/text() = $match/text())
                                )
                            else
                                (true())
                            )
                    )
            " or ./sw:continue
            then
            else
        <xs:annotation>
            <xs:documentation>returns false if any instruction has a repeating equipment, e.g two seperate
equipment elements both containing fins</xs:documentation>
        </xs:annotation>
</xs:assert>
<xs:assert test="
                if (count(./sw:segmentName) > 0)
                    (
                        count(./sw:segmentName/../../../ancestor::* ) = 0
                    )
            " then
            else
        <xs:annotation>
            <xs:documentation>returns false if any segmentName element is not at the top level e.g. it is
nested within a repetition</xs:documentation>
        </xs:annotation>
</xs:assert>
<!-- check if this is just for standard stroke or for kicks and drills too -->
<xs:assert test="
                every $stroke in ./sw:stroke
            satisfies
                if ($stroke/sw:standardStroke = 'individualMedleyOverlap'
            or $stroke/sw:standardStroke = 'individualMedleyOrder' or $stroke/sw:standardStroke =
            'reverseIndividualMedleyOrder' or $stroke/sw:kicking/sw:standardKick =
            'individualMedleyOverlap' or $stroke/sw:kicking/sw:standardKick = 'individualMedleyOrder' or
            $stroke/sw:kicking/sw:standardKick = 'reverseIndividualMedleyOrder' or
            $stroke/sw:drill/sw:drillStroke = 'individualMedleyOverlap' or $stroke/sw:drill/sw:drillStroke =
            'individualMedleyOrder' or $stroke/sw:drill/sw:drillStroke = 'reverseIndividualMedleyOrder'
                then
                    ($stroke/ancestor::*/sw:repetition)
            or
            ($stroke/ancestor::*/sw:continue/sw:continueLength)
            else
                ($stroke/parent::*)
            "
        >

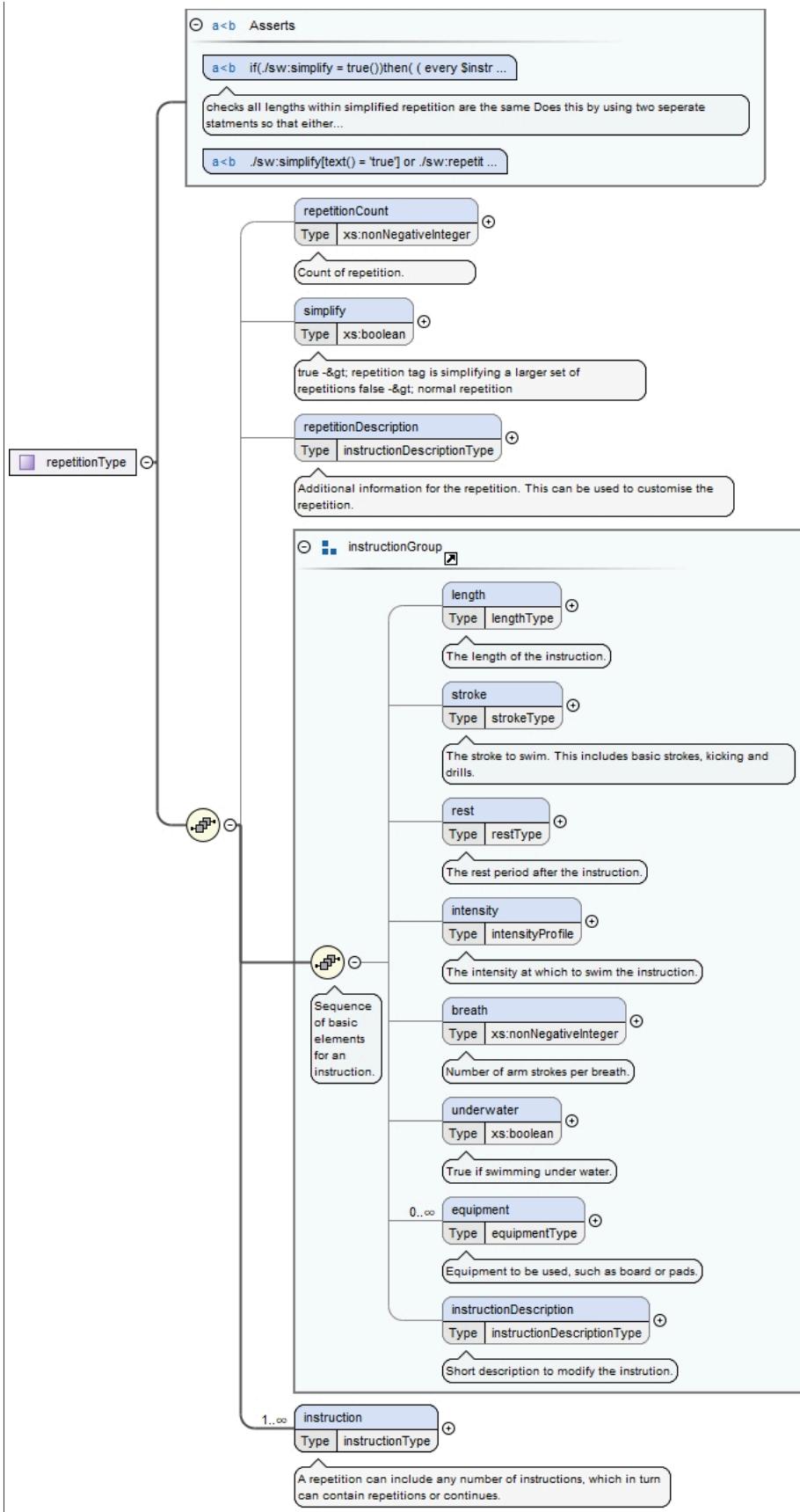
```

```
<xs:annotation>
  <xs:documentation>checks all strokes to make sure medley order or overlaps are only used in a
repetition</xs:documentation>
</xs:annotation>
</xs:assert>
</xs:complexType>
```

Complex Type repetitionType

Namespace	https://github.com/bartneck/swiML
Annotations	

Diagram



Used by

Element `instructionType/repetition`

Model

```
repetitionCount{0,1} , simplify{0,1} , repetitionDescription{0,1} , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} ,
breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1} , instruction+
```

Children	breath, equipment, instruction, instructionDescription, intensity, length, repetitionCount, repetitionDescription, rest, simplify, stroke, underwater	
Asserts	<p>Test</p> <pre>if(.//sw:simplify = true())then((every \$instruction in ./sw:instruction[not(.//sw:pyramid or ./sw:segmentName)] satisfies((if(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) then(if(count(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) = 1) then(number((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1])//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsDistance)) else(sum((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1])//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsDistance)) else(0)+(if(\$instruction/descendant-or-self::sw:continueLength) then(number(\$instruction/descendant-or-self::sw:continueLength)) else(0)) = number(((./descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1])//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsDistance) (./descendant-or-self::sw:instruction[not(ancestor::sw:continueLength)][1])))or(every \$instruction in ./sw:instruction[not(.//sw:pyramid or ./sw:segmentName)] satisfies((if(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) then(if(count(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)]) = 1) then(number((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1])//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsLaps)) else(sum((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1])//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsLaps)) else(0)+(if(\$instruction/descendant-or-self::sw:continueLength) then(number(\$instruction/descendant-or-self::sw:continueLength)) else(0)) = number(((./descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength) and not(.//sw:continue/sw:continueLength) and not(.//sw:repetition)][1])//(preceding-sibling::sw:length ancestor-or-self::*//sw:length)[last()]/sw:lengthAsLaps) (./descendant-or-self::sw:instruction[not(ancestor::sw:continueLength)][1])))else(true())</pre> <p>checks all lengths within simplified repetition are the same Does this by using two separate statements so that either all lengthAsDistances are the same or all lengthAsLaps checks that every instruction, repetition and continue has the same base length e.g. 100m , 4x100m and 50,50 swim as 100 ; these are all of length 100 so can be simplified in the same repetition has to choose whether to use continueLength or add up all instructions within a continue</p> <p>.//sw:simplify[text() = 'true'] or ./sw:repetitionCount and not(.//sw:simplify[text() = 'true'] and ./sw:repetitionCount)</p>	XPath default namespace
Source	<pre><xs:complexType name="repetitionType"> <xs:annotation> <xs:documentation></xs:documentation> </xs:annotation> <xs:sequence> <xs:element name="repetitionCount" type="xs:nonNegativeInteger" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>Count of repetition.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="simplify" minOccurs="0" maxOccurs="1" type="xs:boolean"></pre>	


```

        )
        )
        )else(
        true()
        )

    >
<xs:annotation>
<xs:documentation>checks all lengths within simplified repetition are the same Does this by
using two seperate statements so that either all lengthAsDistances are the same or all lengthAsLaps
checks that every instruction, repetition and continue has the same base length e.g. 100m , 4x100m
and 50,50 swim as 100 ; these are all of length 100 so can be simplified in the same repetition
has to choose whether to use continueLength or add up all instructions within a continue</
<xs:documentation>
</xs:annotation>
</xs:assert>
<!-- Explain what this assertion does -->
<xs:assert test=".//sw:simplify[text() = 'true'] or ./sw:repetitionCount and not(./
sw:simplify[text() = 'true'] and ./sw:repetitionCount)"/>
</xs:complexType>

```

Complex Type lengthType

Namespace	https://github.com/bartneck/swiML
Annotations	The length for a swimming instruction.
Diagram	<pre> classDiagram class lengthType { <<Mixed true>> <<The length for a swimming instruction.>> <<Length can be described as distance or time.>> } lengthType o-- lengthAsDistance lengthType o-- lengthAsTime lengthType o-- lengthAsLaps lengthAsDistance <<Length of instruction as distance.>> lengthAsDistance <<Type xs:nonNegativeInteger>> lengthAsTime <<Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30.>> lengthAsTime <<Type xs:duration>> lengthAsLaps <<Length of instruction in number of laps.>> lengthAsLaps <<Type xs:nonNegativeInteger>> </pre>
Properties	mixed: true
Used by	Elements continueType/continueLength, instructionGroup/length, pyramidType/startLength, pyramidType/stopLength
Model	lengthAsDistance lengthAsTime lengthAsLaps
Children	lengthAsDistance, lengthAsLaps, lengthAsTime
Source	<pre> <xs:complexType name="lengthType" mixed="true"> <xs:annotation> <xs:documentation>The length for a swimming instruction.</xs:documentation> </xs:annotation> <!-- Length as either distance, laps or time --> <xs:choice> <xs:annotation> <xs:documentation>Length can be described as distance or time.</xs:documentation> </xs:annotation> <xs:element name="lengthAsDistance" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>Length of instruction as distance.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="lengthAsTime" type="xs:duration"> <xs:annotation> <xs:documentation>Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="lengthAsLaps" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>Length of instruction in number of laps.</xs:documentation> </xs:annotation> </xs:element> </xs:choice> </xs:complexType> </pre>

Complex Type strokeType

Namespace	https://github.com/bartneck/swiML
Annotations	Stroke types.

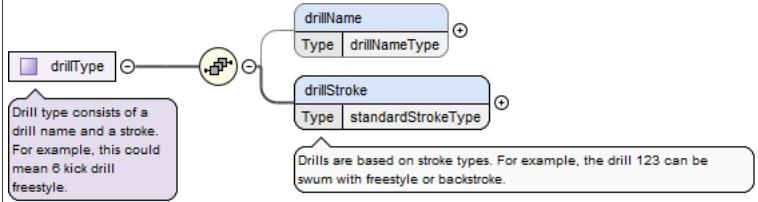
Diagram	
Properties	mixed: true
Used by	Element instructionGroup/stroke
Model	standardStroke kicking drill
Children	drill, kicking, standardStroke
Source	<pre><xs:complexType name="strokeType" mixed="true"> <xs:annotation> <xs:documentation>Stroke types.</xs:documentation> </xs:annotation> <xs:choice> <xs:element name="standardStroke" type="standardStrokeType"/> <xs:element name="kicking" type="kickStyle"/> <xs:element name="drill" type="drillType"/> </xs:choice> </xs:complexType></pre>

Complex Type kickStyle

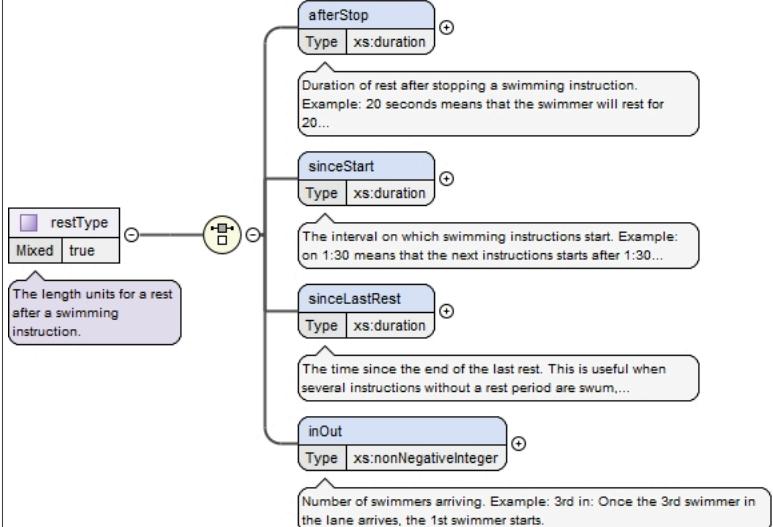
Namespace	https://github.com/bartneck/swiML
Diagram	
Used by	Element strokeType/kicking
Model	(orientation{0,1} , legMovement) standardKick
Children	legMovement, orientation, standardKick
Source	<pre><xs:complexType name="kickStyle"> <xs:choice> <xs:sequence> <xs:element name="orientation" type="orientationType" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>The orientation of the swimmers body.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="legMovement" type="legMovementType" minOccurs="1" maxOccurs="1"> <xs:annotation> <xs:documentation>The style of the leg movements.</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> <xs:element name="standardKick" minOccurs="1" maxOccurs="1" type="standardStrokeType"/> </xs:choice> </xs:complexType></pre>

Complex Type drillType

Namespace	https://github.com/bartneck/swiML
Annotations	Drill type consists of a drill name and a stroke. For example, this could mean 6 kick drill freestyle.

Diagram	
Used by	Element strokeType/drill
Model	drillName{0,1} , drillStroke
Children	drillName, drillStroke
Source	<pre><xss:complexType name="drillType"> <xss:annotation> <xss:documentation>Drill type consists of a drill name and a stroke. For example, this could mean 6 kick drill freestyle.</xss:documentation> </xss:annotation> <xss:sequence> <xss:element name="drillName" minOccurs="0" maxOccurs="1" type="drillNameType"/> <xss:element name="drillStroke" type="standardStrokeType" maxOccurs="1" minOccurs="1"> <xss:annotation> <xss:documentation>Drills are based on stroke types. For example, the drill 123 can be swum with freestyle or backstroke.</xss:documentation> </xss:annotation> </xss:element> </xss:sequence> </xss:complexType></pre>

Complex Type restType

Namespace	https://github.com/bartneck/swiML
Annotations	The length units for a rest after a swimming instruction.
Diagram	
Properties	mixed: true
Used by	Element instructionGroup/rest
Model	afterStop sinceStart sinceLastRest inOut
Children	afterStop, inOut, sinceLastRest, sinceStart
Source	<pre><xss:complexType name="restType" mixed="true"> <xss:annotation> <xss:documentation>The length units for a rest after a swimming instruction.</xss:documentation> </xss:annotation> <xss:choice> <xss:element name="afterStop" type="xs:duration"> <xss:annotation> <xss:documentation>Duration of rest after stopping a swimming instruction. Example: 20 seconds means that the swimmer will rest for 20 seconds after stopping the current instructions.</xss:documentation> </xss:annotation> </xss:element> </xss:choice> </xss:complexType></pre>

```

</xs:element>
<xs:element name="sinceStart" type="xs:duration">
  <xs:annotation>
    <xs:documentation>The interval on which swimming instructions start. Example: on 1:30 means that the next instructions starts after 1:30 from starting the current instruction.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="sinceLastRest" type="xs:duration">
  <xs:annotation>
    <xs:documentation>The time since the end of the last rest. This is useful when several instructions without a rest period are swum, followed by a since start type rest.</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="inOut" type="xs:nonNegativeInteger">
  <xs:annotation>
    <xs:documentation>Number of swimmers arriving. Example: 3rd in: Once the 3rd swimmer in the lane arrives, the 1st swimmer starts.</xs:documentation>
  </xs:annotation>
</xs:element>
</xs:choice>
</xs:complexType>

```

Complex Type intensityProfile

Namespace	https://github.com/bartneck/swiML
Annotations	The intensity of the instruction. When given at the lowest level just start intensity indicates a constant intensity if the stop intensity is given then it is a build within the instruction If the intensity is given at a higher level (repetition or continue) just start intensity is the same constant for all child instructions given a stop intensity then it is descending/ascending over the child instructions
Diagram	<pre> classDiagram class intensityProfile { Mixed true intensityType } class startIntensity { Type intensityType } class stopIntensity { Type intensityType } intensityProfile "o--> startIntensity intensityProfile "o--> stopIntensity note over startIntensity, stopIntensity: The intensity of the instruction. When given at the lowest level just start intensity indicates a constant intensity if... </pre>
Properties	mixed: true
Used by	Element instructionGroup/intensity
Model	startIntensity , stopIntensity{0,1}
Children	startIntensity, stopIntensity
Source	<pre> <xs:complexType name="intensityProfile" mixed="true"> <xs:annotation> <xs:documentation>The intensity of the instruction. When given at the lowest level just start intensity indicates a constant intensity if the stop intensity is given then it is a build within the instruction If the intensity is given at a higher level (repetition or continue) just start intensity is the same constant for all child instructions given a stop intensity then it is descending/ascending over the child instructions</xs:documentation> </xs:annotation> <xs:sequence> <xs:element name="startIntensity" minOccurs="1" maxOccurs="1" type="intensityType"/> <xs:element name="stopIntensity" minOccurs="0" maxOccurs="1" type="intensityType"/> </xs:sequence> </xs:complexType> </pre>

Complex Type intensityType

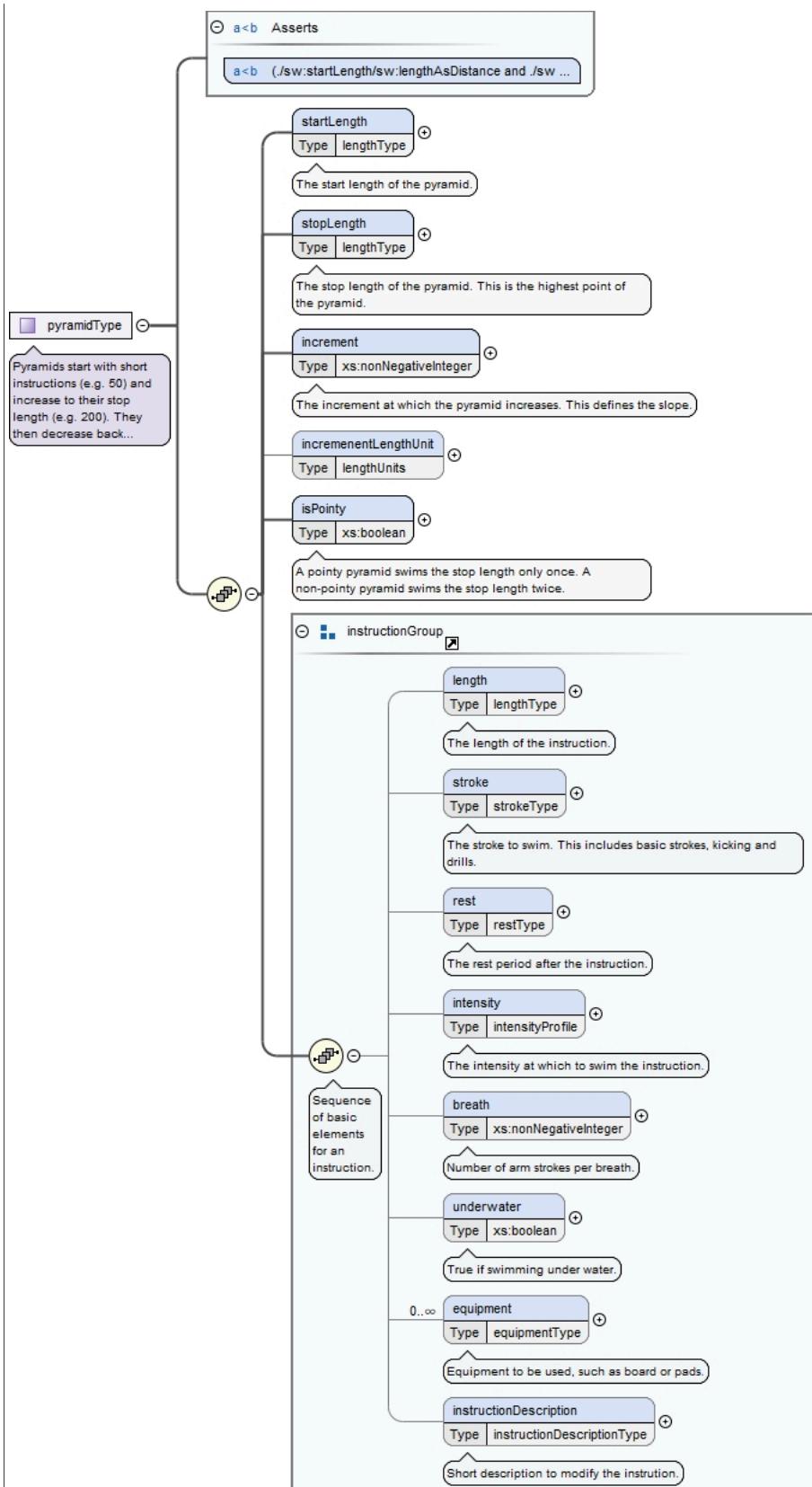
Namespace	https://github.com/bartneck/swiML
Annotations	The intensity of the instructions.

Diagram	<pre> classDiagram class intensityType { <<The intensity of the instructions.>> } class percentageEffort { <<Effort in percentage. Example: 100 means maximum effort.>> } class zone { <<Effort in training zone.>> } class percentageHeartRate { <<Heart rate in percentage of maximum heart rate.>> } intensityType < -- percentageEffort intensityType < -- zone intensityType < -- percentageHeartRate </pre>
Used by	Elements intensityProfile/startIntensity, intensityProfile/stopIntensity
Model	percentageEffort zone percentageHeartRate
Children	percentageEffort, percentageHeartRate, zone
Source	<pre> <xsd:complexType name="intensityType"> <xsd:annotation> <xsd:documentation>The intensity of the instructions.</xsd:documentation> </xsd:annotation> <xsd:choice> <xsd:element name="percentageEffort" type="percentType"> <xsd:annotation> <xsd:documentation>Effort in percentage. Example: 100 means maximum effort.</xsd:documentation> </xsd:annotation> </xsd:element> <xsd:element name="zone" type="zoneType"> <xsd:annotation> <xsd:documentation>Effort in training zone.</xsd:documentation> </xsd:annotation> </xsd:element> <xsd:element name="percentageHeartRate" type="percentType"> <xsd:annotation> <xsd:documentation>Heart rate in percentage of maximum heart rate.</xsd:documentation> </xsd:annotation> </xsd:element> </xsd:choice> </xsd:complexType> </pre>

Complex Type pyramidType

Namespace	https://github.com/bartneck/swiML
Annotations	Pyramids start with short instructions (e.g. 50) and increase to their stop length (e.g. 200). They then decrease back to the start length.

Diagram



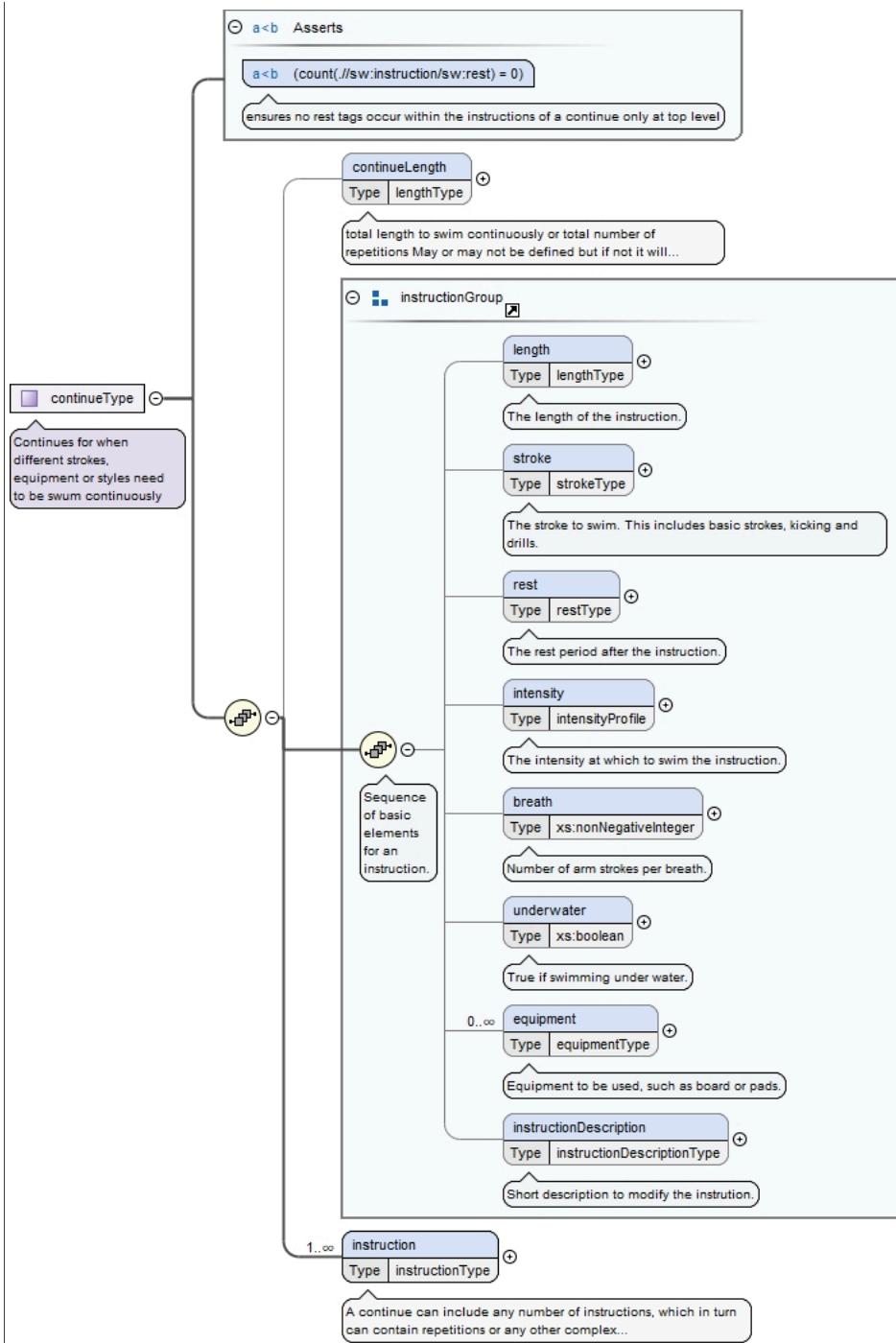
Used by	Element <i>instructionType/pyramid</i>
Model	<i>startLength , stopLength , increment , incremenentLengthUnit{0,1} , isPointy , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1}</i>
Children	<i>breath, equipment, incremenentLengthUnit, increment, instructionDescription, intensity, isPointy, length, rest, startLength, stopLength, stroke, underwater</i>

Asserts	Test	XPath default namespace
	(./sw:startLength/sw:lengthAsDistance and ./sw:stopLength/sw:lengthAsDistance) or (./sw:startLength/sw:lengthAsLaps and ./sw:stopLength/sw:lengthAsLaps) or (./sw:startLength/sw:lengthAsTime and ./sw:stopLength/sw:lengthAsTime)	
Source	<pre> <xs:complexType name="pyramidType"> <xs:annotation> <xs:documentation>Pyramids start with short instructions (e.g. 50) and increase to their stop length (e.g. 200). They then decrease back to the start length.</xs:documentation> </xs:annotation> <xs:sequence> <xs:element name="startLength" minOccurs="1" maxOccurs="1" type="lengthType"> <xs:annotation> <xs:documentation>The start length of the pyramid.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="stopLength" minOccurs="1" maxOccurs="1" type="lengthType"> <xs:annotation> <xs:documentation>The stop length of the pyramid. This is the highest point of the pyramid.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="increment" minOccurs="1" maxOccurs="1" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The increment at which the pyramid increases. This defines the slope.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="incrementLengthUnit" type="lengthUnits" minOccurs="0" maxOccurs="1"/> <xs:element name="isPointy" minOccurs="1" maxOccurs="1" type="xs:boolean"> <xs:annotation> <xs:documentation>A pointy pyramid swims the stop length only once. A non-pointy pyramid swims the stop length twice.</xs:documentation> </xs:annotation> </xs:element> <xs:group ref="instructionGroup"/> </xs:sequence> <xs:assert test=" (./sw:startLength/sw:lengthAsDistance and ./sw:stopLength/ or (./sw:startLength/sw:lengthAsLaps and ./sw:stopLength/ or (./sw:startLength/sw:lengthAsTime and ./sw:stopLength/ " /> </xs:complexType></pre>	

Complex Type continueType

Namespace	https://github.com/bartneck/swiML
Annotations	Continues for when different strokes, equipment or styles need to be swum continuously

Diagram



Used by	Element	instructionType/continue
Model		continueLength{0,1} , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1} , instruction+
Children		breath, continueLength, equipment, instruction, instructionDescription, intensity, length, rest, stroke, underwater
Asserts	Test <code>(count(./sw:instruction/sw:rest) = 0)</code>	XPath default namespace
		ensures no rest tags occur within the instructions of a continue only at top level
Source	<pre> <xss:complexType name="continueType"> <xss:annotation> <xss:documentation>Continues for when different strokes, equipment or styles need to be swum continuously</xss:documentation> </xss:annotation> <xss:sequence> </pre>	

```

<xs:element name="continueLength" minOccurs="0" maxOccurs="1" type="lengthType">
    <xs:annotation>
        <xs:documentation>total length to swim continuously or total number of repetitions May
        or may not be defined but if not it will automatically calculated from given instructions</
    xs:documentation>
    </xs:annotation>
</xs:element>
<!-- Common elements for instructions -->
<xs:group ref="instructionGroup"/>
<xs:element name="instruction" minOccurs="1" maxOccurs="unbounded" type="instructionType">
    <xs:annotation>
        <xs:documentation>A continue can include any number of instructions, which in turn can
        contain repetitions or any other complex instruction type.</xs:documentation>
    </xs:annotation>
    <xs:unique name="contEquipmentUnique">
        <xs:annotation>
            <xs:documentation>Ensures all equipment values in an instruction are unique</
        xs:documentation>
        </xs:annotation>
        <xs:selector xpath=".//sw:equipment" />
        <xs:field xpath="./*" />
    </xs:unique>
    </xs:element>
</xs:sequence>
<!-- ===== -->
<!-- Assertions -->
<!-- Explain what this assertion does -->
<xs:assert test="(count(.//sw:instruction/sw:rest) = 0)">
    <xs:annotation>
        <xs:documentation>ensures no rest tags occur within the instructions of a continue only at top
        level</xs:documentation>
    </xs:annotation>
</xs:assert>
</xs:complexType>

```

Element Group(s)

Element Group instructionGroup

Namespace	https://github.com/bartneck/swiML
Diagram	<p>The diagram illustrates the structure of the <code>instructionGroup</code> element group. It consists of a sequence of basic elements:</p> <ul style="list-style-type: none"> length: Type <code>lengthType</code>. Description: The length of the instruction. stroke: Type <code>strokeType</code>. Description: The stroke to swim. This includes basic strokes, kicking and drills. rest: Type <code>restType</code>. Description: The rest period after the instruction. intensity: Type <code>intensityProfile</code>. Description: The intensity at which to swim the instruction. breath: Type <code>xs:nonNegativeInteger</code>. Description: Number of arm strokes per breath. underwater: Type <code>xs:boolean</code>. Description: True if swimming under water. equipment: Type <code>equipmentType</code>. Description: Equipment to be used, such as board or pads. instructionDescription: Type <code>instructionDescriptionType</code>. Description: Short description to modify the instruction. <p>A note indicates that the sequence contains "Sequence of basic elements for an instruction."</p>

Used by	Complex Types continueType, instructionType, pyramidType, repetitionType
Model	length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1}
Children	breath, equipment, instructionDescription, intensity, length, rest, stroke, underwater
Source	<pre> <xs:group name="instructionGroup"> <xs:sequence> <xs:annotation> <xs:documentation>Sequence of basic elements for an instruction.</xs:documentation> </xs:annotation> <xs:element name="length" minOccurs="0" maxOccurs="1" type="lengthType"> <xs:annotation> <xs:documentation>The length of the instruction.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="stroke" minOccurs="0" maxOccurs="1" type="strokeType"> <xs:annotation> <xs:documentation>The stroke to swim. This includes basic strokes, kicking and drills.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="rest" minOccurs="0" maxOccurs="1" type="restType"> <xs:annotation> <xs:documentation>The rest period after the instruction.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="intensity" minOccurs="0" maxOccurs="1" type="intensityProfile"> <xs:annotation> <xs:documentation>The intensity at which to swim the instruction.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="breath" minOccurs="0" maxOccurs="1" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>Number of arm strokes per breath.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="underwater" minOccurs="0" maxOccurs="1" type="xs:boolean"> <xs:annotation> <xs:documentation>True if swimming under water.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="equipment" minOccurs="0" maxOccurs="unbounded" type="equipmentType"> <xs:annotation> <xs:documentation>Equipment to be used, such as board or pads.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="instructionDescription" type="instructionDescriptionType" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>Short description to modify the instruction.</xs:documentation> </xs:annotation> </xs:element> </xs:sequence> </xs:group></pre>