

Schema documentation for swiML.xsd

may 10, 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	5
Element program / author / firstName	6
Element program / author / lastName	7
Element program / author / email	7
Element program / programDescription	7
Element program / creationDate	7
Element program / poolLength	8
Element program / lengthUnit	8
Element program / programLength	9
Element program / hideIntro	9
Element program / layoutWidth	9
Element program / instruction	10
Element instructionType / segmentName	13
Element instructionType / repetition	13
Element repetitionType / repetitionCount	15
Element repetitionType / simplify	16
Element repetitionType / repetitionDescription	16
Element instructionGroup / length	16
Element lengthType / lengthAsDistance	17
Element lengthType / lengthAsTime	17
Element lengthType / lengthAsLaps	18
Element instructionGroup / stroke	18
Element strokeType / standardStroke	18
Element strokeType / kicking	19
Element kickStyle / orientation	19
Element kickStyle / legMovement	20
Element kickStyle / standardKick	20
Element strokeType / drill	21
Element drillType / drillName	21
Element drillType / drillStroke	22
Element instructionGroup / rest	23
Element restType / afterStop	23
Element restType / sinceStart	24
Element restType / sinceLastRest	24
Element restType / inOut	24
Element instructionGroup / intensity	25
Element intensityProfile / startIntensity	25
Element intensityType / percentageEffort	26
Element intensityType / zone	26
Element intensityType / percentageHeartRate	26
Element intensityProfile / stopIntensity	27
Element instructionGroup / breath	27
Element instructionGroup / underwater	28
Element instructionGroup / equipment	28
Element instructionGroup / instructionDescription	28
Element instructionGroup / excludeAlign	29
Element repetitionType / instruction	29
Element instructionType / pyramid	32
Element pyramidType / startLength	34
Element pyramidType / startlengthUnit	34
Element pyramidType / stopLength	35
Element pyramidType / stoplengthUnit	35
Element pyramidType / increment	35
Element pyramidType / incremenentLengthUnit	36
Element pyramidType / isPointy	36
Element pyramidType / excludeAlignPyramid	36

Element instructionType / continue	37
Element continueType / continueLength	38
Element continueType / instruction	38
Element continueType / excludeAlignContinue	41
Element repetitionType / excludeAlignRepetition	41
Simple Type(s)	41
Simple Type titleString	41
Simple Type emailAddress	42
Simple Type descriptionString	42
Simple Type lengthUnits	42
Simple Type segmentNameType	43
Simple Type instructionDescriptionType	43
Simple Type standardStrokeType	43
Simple Type orientationType	44
Simple Type legMovementType	45
Simple Type drillNameType	45
Simple Type percentType	46
Simple Type zoneType	46
Simple Type equipmentType	47
Simple Type lengthUnitType	47
Simple Type equipmentList	47
Complex Type(s)	48
Complex Type instructionType	48
Complex Type repetitionType	50
Complex Type lengthType	53
Complex Type strokeType	54
Complex Type kickStyle	54
Complex Type drillType	55
Complex Type restType	55
Complex Type intensityProfile	56
Complex Type intensityType	56
Complex Type pyramidType	57
Complex Type continueType	59
Element Group(s)	61
Element Group instructionGroup	61

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

Schema(s)

Main schema **swiML.xsd**

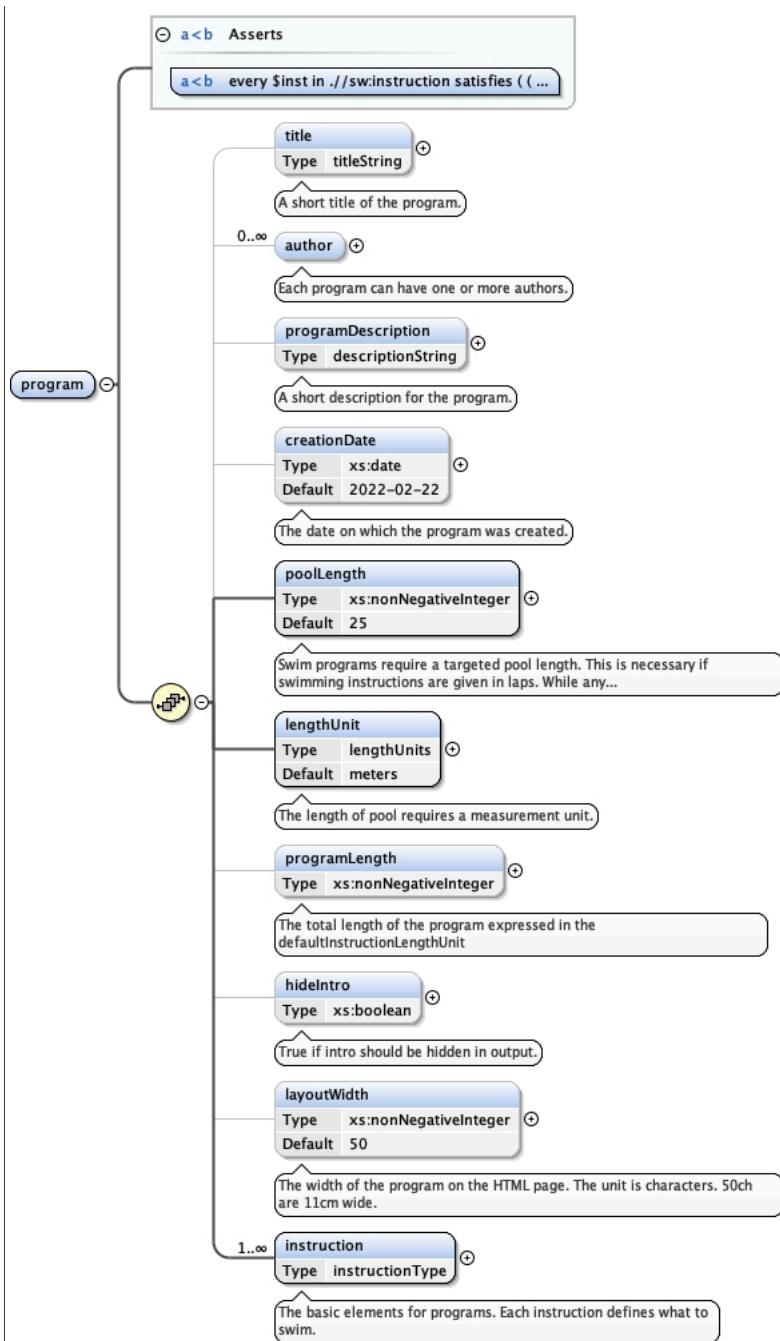
Namespace	https://github.com/bartneck/swiML
Properties	attribute form default: unqualified element form default: qualified version: 2.2

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 , programLength{0,1} , hideIntro{0,1} , layoutWidth{0,1} , instruction+
Children	author, creationDate, hideIntro, instruction, layoutWidth, lengthUnit, poolLength, programDescription, programLength, title
Instance	<pre><program xmlns="https://github.com/bartneck/swiML"> <title>{0,1}</title> <author>{0,unbounded}</author> <programDescription>{0,1}</programDescription> <creationDate>{0,1}</creationDate> <poolLength>{1,1}</poolLength> <lengthUnit>{1,1}</lengthUnit> <programLength>{0,1}</programLength> <hideIntro>{0,1}</hideIntro> <layoutWidth>{0,1}</layoutWidth> <instruction>{1,unbounded}</instruction> </program></pre>

Asserts	Test	XPath default namespace
	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)	
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 minOccurs="1" name="poolLength" type="xs:nonNegativeInteger" maxOccurs="1" default="25"> <xs:annotation> <xs:documentation>Swim programs require a targeted pool length. This is necessary if swimming instructions are given in laps. While any swim program defined for a 50 meter pool can be swum in a 25 meter pool, the reverse cannot be guaranteed.</xs:documentation> </xs:annotation> </xs:element> <xs:element minOccurs="1" name="lengthUnit" type="lengthUnits" maxOccurs="1" default="meters"> <xs:annotation> <xs:documentation>The length of pool requires a measurement unit.</xs:documentation> </xs:annotation> </xs:element> <!-- The total length of the program. --> <xs:element name="programLength" minOccurs="0" maxOccurs="1" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The total length of the program expressed in the defaultInstructionLengthUnit</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> </xs:sequence> </xs:complexType> </pre>	

```

        </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>
<!-- ===== -->
<!-- Assertion -->
<!-- checks every instruction has stroke, rest and length defined  
any other element in an instruction doesnt have to be defined  
for some reason adding this makes it work?-->
<!-- ===== -->
<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)" />
</xs:complexType>
</xs:element>

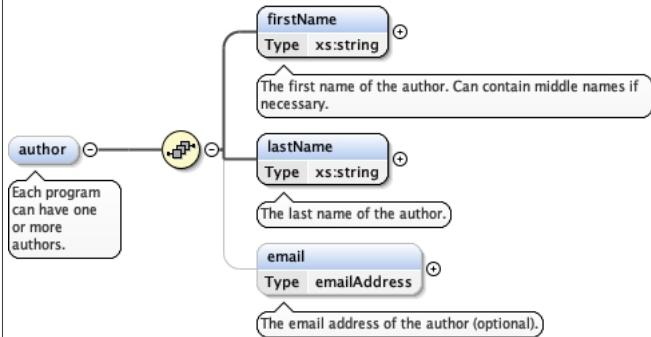
```

Element program / title

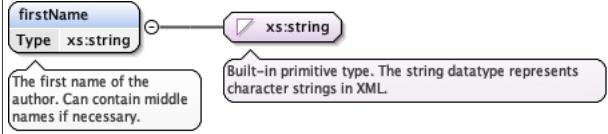
Namespace	https://github.com/bartneck/swiML						
Annotations	A short title of the program.						
Diagram	<pre> classDiagram class title { titleString } title "A short title of the program." title "*" "The length of the title is constraint in length." </pre>						
Type	titleString						
Properties	<table 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> </xs:element> </pre>						

Element program / author

Namespace	https://github.com/bartneck/swiML
Annotations	Each program can have one or more authors.

Diagram	
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: unbounded</p>
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	
Type	xs:string
Properties	<p>content: simple</p> <p>minOccurs: 1</p>
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	<pre> graph LR lastName[lastName Type xs:string] --> xsString[xs:string] subgraph Callout direction TB C1["The last name of the author."] C2["Built-in primitive type. The string datatype represents character strings in XML."] C1 --- C2 end </pre>				
Type	xs:string				
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> </table>	content:	simple	minOccurs:	1
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	<pre> graph LR email[email Type emailAddress] --> emailAddress[emailAddress] subgraph Callout direction TB C1["The email address of the author (optional)."] C2["The pattern checks for valid email addresses."] C1 --- C2 end </pre>				
Type	emailAddress				
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> </table>	content:	simple	minOccurs:	0
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	<pre> graph LR programDescription[programDescription Type descriptionString] --> descriptionString[descriptionString] subgraph Callout direction TB C1["A short description for the program."] C2["The length of the description text is constraint in length."] C1 --- C2 end </pre>						
Type	descriptionString						
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 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	<pre> classDiagram class creationDate { Type xs:date Default 2022-02-22 } xs:date < -- creationDate note over creationDate: The date on which the program was created. note over xs:date: Built-in primitive type. The date datatype represents a calendar date. </pre>
Type	xs:date
Properties	content: simple minOccurs: 0 maxOccurs: 1 default: 2022-02-22
Source	<pre> <xss:element minOccurs="0" name="creationDate" type="xs:date" maxOccurs="1" default="2022-02-22"> <xss:annotation> <xss:documentation>The date on which the program was created.</xss:documentation> </xss:annotation> </xss:element> </pre>

Element program / poolLength

Namespace	https://github.com/bartneck/swiML
Annotations	Swim programs require a targeted pool length. This is necessary if swimming instructions are given in laps. While any swim program defined for a 50 meter pool can be swum in a 25 meter pool, the reverse cannot be guaranteed.
Diagram	<pre> classDiagram class poolLength { Type xs:nonNegativeInteger Default 25 } xs:nonNegativeInteger < -- poolLength note over poolLength: Swim programs require a targeted pool length. This is necessary if swimming instructions are given in laps. While any... 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	content: simple minOccurs: 1 maxOccurs: 1 default: 25
Source	<pre> <xss:element minOccurs="1" name="poolLength" type="xs:nonNegativeInteger" maxOccurs="1" default="25"> <xss:annotation> <xss:documentation>Swim programs require a targeted pool length. This is necessary if swimming instructions are given in laps. While any swim program defined for a 50 meter pool can be swum in a 25 meter pool, the reverse cannot be guaranteed.</xss:documentation> </xss:annotation> </xss:element> </pre>

Element program / lengthUnit

Namespace	https://github.com/bartneck/swiML
Annotations	The length of pool requires a measurement unit.
Diagram	<pre> classDiagram class lengthUnit { Type lengthUnits Default meters } lengthUnits < -- lengthUnit note over lengthUnit: The length of pool requires a measurement unit. note over lengthUnits: The unit of measurement for the length of the target pool (meter or yards). </pre>
Type	lengthUnits
Properties	content: simple minOccurs: 1 maxOccurs: 1 default: meters

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

Element program / programLength

Namespace	https://github.com/bartneck/swiML							
Annotations	The total length of the program expressed in the defaultInstructionLengthUnit							
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="programLength" minOccurs="0" maxOccurs="1" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The total length of the program expressed in the defaultInstructionLengthUnit</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								
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="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></pre>							

Element program / layoutWidth

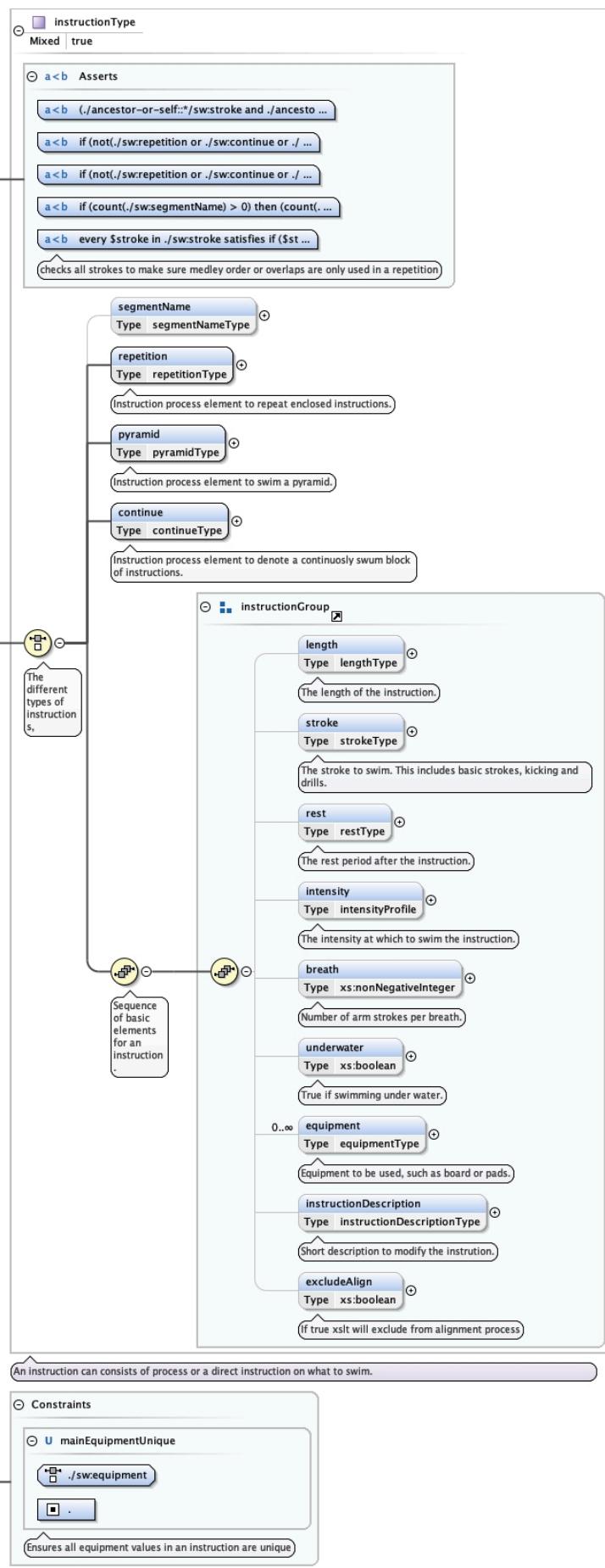
Namespace	https://github.com/bartneck/swiML			
Annotations	The width of the program on the HTML page. The unit is characters. 50ch are 11cm wide.			
Diagram				
Type	xs:nonNegativeInteger			
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> </table>		content:	simple
content:	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> <p>(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName</p> <p>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())</p> <p>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())</p> <p>if (count(.//sw:segmentName) > 0) then (count(.//sw:segmentName//../*) = 0) else (true())</p> <p>every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder') then (\$stroke/parent::*//parent:sw:repetition) else (\$stroke/parent::*//parent:sw:repetition)</p> <p>checks all strokes to make sure medley order or overlaps are only used in a repetition</p>	<p>XPath default namespace</p>										
Identity constraints	<table border="1"> <thead> <tr> <th>QName</th><th>Type</th><th>Refer</th><th>Selector</th><th>Field(s)</th></tr> </thead> <tbody> <tr> <td>mainEquipmentUnique</td><td>unique</td><td></td><td>/sw:equipment</td><td>.</td></tr> </tbody> </table>	QName	Type	Refer	Selector	Field(s)	mainEquipmentUnique	unique		/sw:equipment	.	
QName	Type	Refer	Selector	Field(s)								
mainEquipmentUnique	unique		/sw:equipment	.								
Source	<pre><xss:element name="instruction" type="instructionType" minOccurs="1" maxOccurs="unbounded"> <xss:annotation> <xss:documentation>The basic elements for programs. Each instruction defines what to swim.</xss:documentation> </xss:annotation> <xss:unique name="mainEquipmentUnique"> <xss:annotation> <xss:documentation>Ensures all equipment values in an instruction are unique</xss:documentation> </xss:annotation> <xss:selector xpath=".//sw:equipment"/> <xss:field xpath=".//sw:equipment"/> </xss:unique> </xss:element></pre>											

<pre></xs:unique> </xs:element></pre>

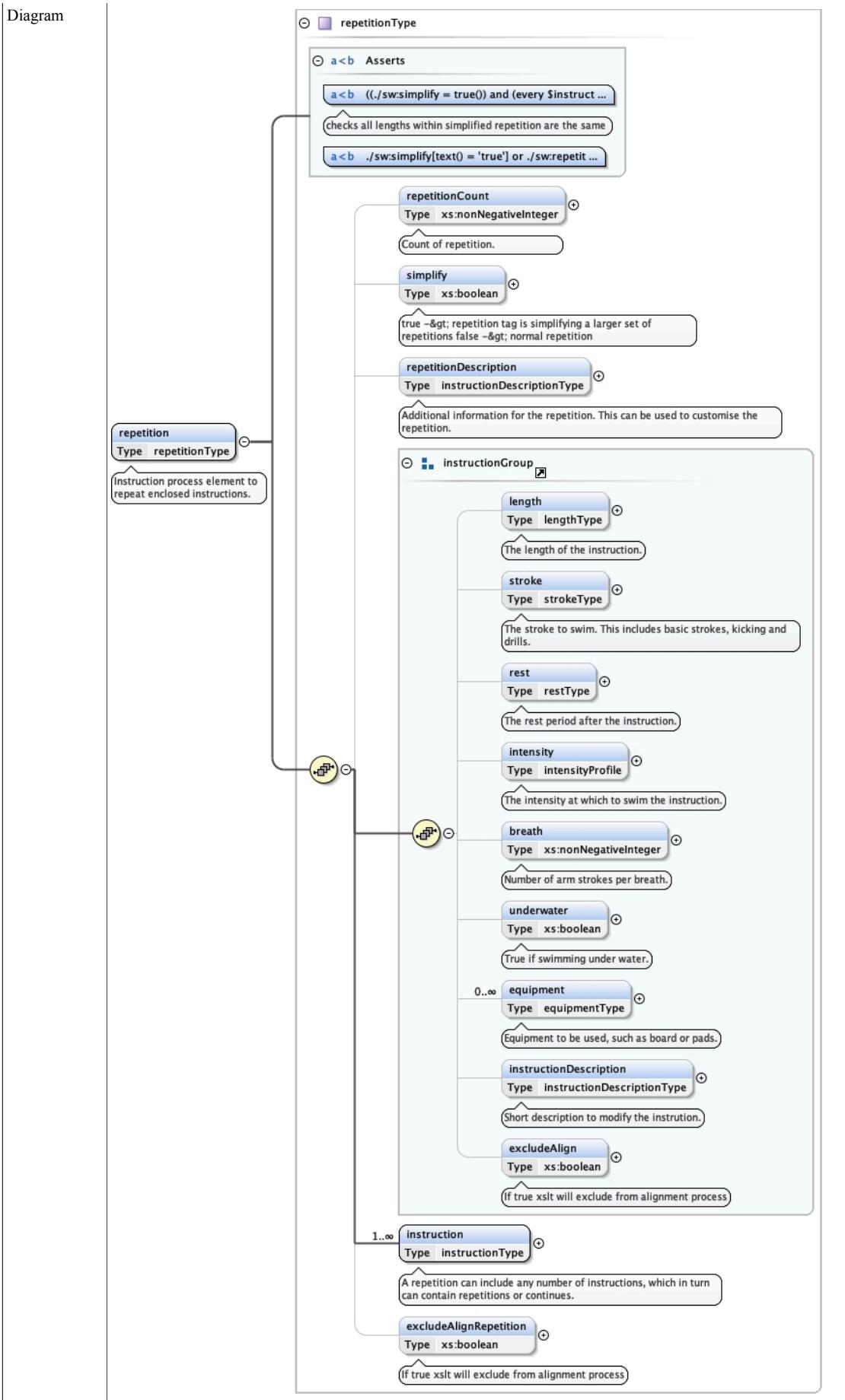
Element instructionType / segmentName

Namespace	https://github.com/bartneck/swiML	
Diagram		
Type	segmentNameType	
Properties	content: simple minOccurs: 0 maxOccurs: 1	
Facets	maxLength	30
Source	<code><xs:element name="segmentName" minOccurs="0" maxOccurs="1" type="segmentNameType"/></code>	

Element instructionType / repetition

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

Diagram



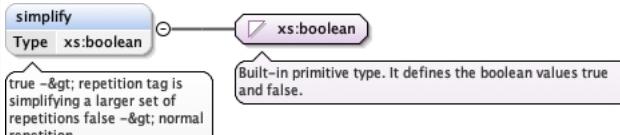
Type	repetitionType	
Properties	content: complex	
Model	repetitionCount{0,1} , simplify{0,1} , repetitionDescription{0,1} , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1} , excludeAlign{0,1} , instruction+ , excludeAlignRepetition{0,1}	
Children	breathe, equipment, excludeAlign, excludeAlignRepetition, 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> <excludeAlign>{0,1}</excludeAlign> <instruction>{1,unbounded}</instruction> <excludeAlignRepetition>{0,1}</excludeAlignRepetition> </repetition></pre>	
Asserts	<p>Test</p> <pre>((./sw:simplify = true()) and (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)][not(.//sw:continue/sw:continueLength)]) then(if(count(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(.//sw:continue/sw:continueLength)]) = 1) then(number((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(.//sw:continue/sw:continueLength)][1])//sw:lengthAsDistance)) else(sum((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(.//sw:continue/sw:continueLength)][1])//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)][not(.//sw:continue/sw:continueLength)][1])//sw:lengthAsDistance) ./descendant-or-self::sw:continueLength[1])))) or ./sw:simplify = false() or count(.//sw:simplify) = 0</pre> <p>checks all lengths within simplified repetition are the same</p> <p>./sw:simplify[text() = 'true'] or ./sw:repetitionCount and not(.//sw:simplify[text() = 'true'] and ./sw:repetitionCount)</p>	XPath default namespace
Source	<pre><xss:element name="repetition" type="repetitionType"> <xss:annotation> <xss:documentation>Instruction process element to repeat enclosed instructions.</xss:documentation> </xss:annotation> </xss:element></pre>	

Element repetitionType / repetitionCount

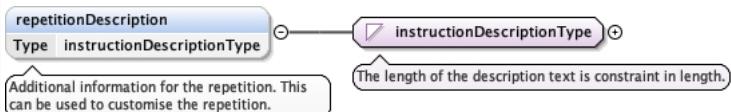
Namespace	https://github.com/bartneck/swiML				
Annotations	Count of repetition.				
Diagram					
Type	xs:nonNegativeInteger				
Properties	<table border="1"> <tr> <td>content:</td> <td>simple</td> </tr> <tr> <td>minOccurs:</td> <td>0</td> </tr> </table>	content:	simple	minOccurs:	0
content:	simple				
minOccurs:	0				

	maxOccurs:	1
Source	<xs:element name="repetitionCount" type="xs:nonNegativeInteger" minOccurs="0" maxOccurs="1"> <xs:annotation> <xs:documentation>Count of repetition.</xs:documentation> </xs:annotation> </xs:element>	

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	content: simple minOccurs: 0 maxOccurs: 1
Source	<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>

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	content: simple minOccurs: 0 maxOccurs: 1
Facets	maxLength 100
Source	<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>

Element instructionGroup / length

Namespace	https://github.com/bartneck/swiML
Annotations	The length of the instruction.

Diagram	<pre> classDiagram lengthType < -- length lengthType < -- lengthAsDistance lengthType < -- lengthAsTime lengthType < -- lengthAsLaps lengthAsDistance < -- lengthAsDistance lengthAsTime < -- lengthAsTime lengthAsLaps < -- lengthAsLaps </pre> <p>The length for a swimming instruction.</p>								
Type	lengthType								
Properties	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 2px;">content:</td><td style="padding: 2px;">complex</td></tr> <tr> <td style="padding: 2px;">minOccurs:</td><td style="padding: 2px;">0</td></tr> <tr> <td style="padding: 2px;">maxOccurs:</td><td style="padding: 2px;">1</td></tr> <tr> <td style="padding: 2px;">mixed:</td><td style="padding: 2px;">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 lengthAsDistance < -- xs:nonNegativeInteger </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	content: simple
Source	<pre> <xs:element name="lengthAsDistance" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>Length of instruction as distance.</xs:documentation> </xs:annotation> </xs:element> </pre>

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	<pre> classDiagram lengthAsTime < -- xs:duration </pre> <p>Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30.</p> <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

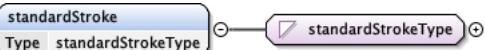
Namespace	https://github.com/bartneck/swiML
Annotations	Length of instruction in number of laps.
Diagram	<p>lengthAsLaps Type xs:nonNegativeInteger</p> <p>Length of instruction in number of laps.</p> <p>Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value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

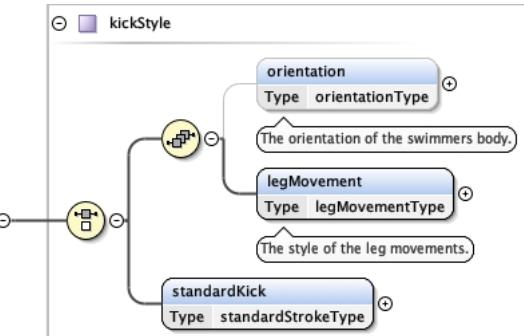
Namespace	https://github.com/bartneck/swiML
Annotations	The stroke to swim. This includes basic strokes, kicking and drills.
Diagram	<p>stroke Type strokeType</p> <p>The stroke to swim. This includes basic strokes, kicking and drills.</p> <p>strokeType Mixed true</p> <p>standardStroke Type standardStrokeType</p> <p>kicking Type kickStyle</p> <p>drill Type drillType</p> <p>Stroke types.</p>
Type	strokeType
Properties	<p>content: complex</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p> <p>mixed: true</p>
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	
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	<xs:element name="standardStroke" type="standardStrokeType" />

Element strokeType / kicking

Namespace	https://github.com/bartneck/swiML
Diagram	
Type	kickStyle
Properties	content: complex
Model	(orientation{0,1} , legMovement) standardKick
Children	legMovement, orientation, standardKick
Instance	<pre> <kicking xmlns="https://github.com/bartneck/swiML"> <orientation>{0,1}</orientation> <legMovement>{1,1}</legMovement> <standardKick>{1,1}</standardKick> </kicking> </pre>
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	<p>content: simple</p> <p>minOccurs: 0</p> <p>maxOccurs: 1</p>
Facets	<p>enumeration front</p> <p>enumeration back</p> <p>enumeration left</p> <p>enumeration right</p> <p>enumeration side</p> <p>enumeration vertical</p> <p>enumeration waka</p>
Source	<pre><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></pre>

Element kickStyle / legMovement

Namespace	https://github.com/bartneck/swiML
Annotations	The style of the leg movements.
Diagram	
Type	legMovementType
Properties	<p>content: simple</p> <p>minOccurs: 1</p> <p>maxOccurs: 1</p>
Facets	<p>enumeration flutter</p> <p>enumeration dolpine</p> <p>enumeration scissor</p>
Source	<pre><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></pre>

Element kickStyle / standardKick

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

	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	<xss:element name="standardKick" minOccurs="1" maxOccurs="1" type="standardStrokeType"/>	

Element strokeType / drill

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

Element drillType / drillName

Namespace	https://github.com/bartneck/swiML
Diagram	<pre> classDiagram class drillName { <<drillNameType>> } class drillNameType { <<drillName>> } drillName "1..1" -- "1..1" drillNameType </pre> <p>Drill names.</p>
Type	drillNameType
Properties	content: simple minOccurs: 1 maxOccurs: 1
Facets	enumeration 6KickDrill enumeration 8KickDrill

	enumeration	10KickDrill
	enumeration	12KickDrill
	enumeration	fingerTrails
	enumeration	fingerDrag
	enumeration	123
	enumeration	bigDog
	enumeration	scull
	enumeration	singleArm
	enumeration	any
	enumeration	technic
	enumeration	dogPaddle
	enumeration	tarzan
	enumeration	6666
	enumeration	3Kick1Pull
	enumeration	other
Source	<xs:element name="drillName" minOccurs="1" 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 drillStroke "1" -- "0..1" standardStrokeType </pre> <p>Drills are based on stroke types. For example, the drill 123 can be swum with freestyle or backstroke.</p>	
Type	standardStrokeType	
Properties	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	<xs:element name="drillStroke" type="standardStrokeType" maxOccurs="1" minOccurs="1"> <xs:annotation>	

```

<xs:documentation>Drills are based on stroke types. For example, the drill 123 can be swum with
freestyle or backstroke.</xs:documentation>
<xs:annotation>
</xs:annotation>

```

Element instructionGroup / rest

Namespace	https://github.com/bartneck/swiML								
Annotations	The rest period after the instruction.								
Diagram	<p>The diagram illustrates the <code>restType</code> element structure. It is a mixed type (<code>Mixed true</code>) containing four attributes:</p> <ul style="list-style-type: none"> <code>afterStop</code>: Type <code>xs:duration</code>. Description: Duration of rest after stopping a swimming instruction. Example: 20 seconds means that the swimmer will rest for 20... <code>sinceStart</code>: Type <code>xs:duration</code>. Description: The interval on which swimming instructions start. Example: on 1:30 means that the next instructions starts after 1:30... <code>sinceLastRest</code>: Type <code>xs:duration</code>. Description: The time since the end of the last rest. This is useful when several instructions without a rest period are swum,... <code>inOut</code>: Type <code>xs:nonNegativeInteger</code>. 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	<code>restType</code>								
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	<code>afterStop sinceStart sinceLastRest inOut</code>								
Children	<code>afterStop, inOut, sinceLastRest, sinceStart</code>								
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> <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> </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	<p>The diagram shows the <code>afterStop</code> attribute of the <code>restType</code> element. It is of type <code>xs:duration</code>. A note indicates: Duration of rest after stopping a swimming instruction. Example: 20 seconds means that the swimmer will rest for 20... Another note states: Built-in primitive type. The duration datatype represents a duration of time.</p>

Type	xs:duration
Properties	content: simple
Source	<pre><xs:element name="afterStop" type="xs:duration"> <xs:annotation> <xs: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.</xs:documentation> </xs:annotation> </xs: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><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></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	
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	
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	<pre> classDiagram class intensityProfile { startIntensity : intensityType stopIntensity : intensityType } intensityProfile < -- intensity intensity < -- intensityProfile </pre> <p>The diagram shows the <code>intensityProfile</code> element as a class with two attributes: <code>startIntensity</code> and <code>stopIntensity</code>. Both attributes are of type <code>intensityType</code>. A note below the <code>intensityProfile</code> class states: "The intensity of the instruction. When given at the lowest level just start intensity indicates a constant intensity if..."</p>								
Type	intensityProfile								
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	startIntensity , stopIntensity{0,1}								
Children	startIntensity, stopIntensity								
Instance	<pre><intensity xmlns="https://github.com/bartneck/swiML"> <startIntensity>{1,1}</startIntensity> <stopIntensity>{0,1}</stopIntensity> </intensity></pre>								
Source	<pre><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></pre>								

Element intensityProfile / startIntensity

Namespace	https://github.com/bartneck/swiML				
Diagram	<pre> classDiagram class intensityType { percentageEffort : percentType zone : zoneType percentageHeartRate : percentType } intensityType < -- startIntensity </pre> <p>The diagram shows the <code>intensityType</code> element as a class with three subclasses: <code>percentageEffort</code>, <code>zone</code>, and <code>percentageHeartRate</code>, all of type <code>percentType</code>. A note below the <code>intensityType</code> class states: "The intensity of the instructions."</p>				
Type	intensityType				
Properties	<table border="1"> <tr> <td>content:</td> <td>complex</td> </tr> <tr> <td>minOccurs:</td> <td>1</td> </tr> </table>	content:	complex	minOccurs:	1
content:	complex				
minOccurs:	1				

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

Element intensityType / percentageEffort

Namespace	https://github.com/bartneck/swiML					
Annotations	Effort in percentage. Example: 100 means maximum effort.					
Diagram						
Type	percentType					
Properties	content: simple					
Facets	<table> <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	<xss:element name="percentageEffort" type="percentType"> <xss:annotation> <xss:documentation>Effort in percentage. Example: 100 means maximum effort.</xss:documentation> </xss:annotation> </xss:element>					

Element intensityType / zone

Namespace	https://github.com/bartneck/swiML											
Annotations	Effort in training zone.											
Diagram												
Type	zoneType											
Properties	content: simple											
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											
Source	<xss:element name="zone" type="zoneType"> <xss:annotation> <xss:documentation>Effort in training zone.</xss:documentation> </xss:annotation> </xss:element>											

Element intensityType / percentageHeartRate

Namespace	https://github.com/bartneck/swiML	
Annotations	Heart rate in percentage of maximum heart rate.	
Diagram		

Type	percentType
Properties	content: simple
Facets	maxInclusive 100 minInclusive 0
Source	<pre><xs:element name="percentageHeartRate" type="percentType"> <xs:annotation> <xs:documentation>Heart rate in percentage of maximum heart rate.</xs:documentation> </xs:annotation> </xs:element></pre>

Element intensityProfile / stopIntensity

Namespace	https://github.com/bartneck/swiML
Diagram	<pre> classDiagram class stopIntensity { <<intensityType>> <<complex>> <<0..1>> <<1>> } class percentageEffort { <<percentType>> <<100 means maximum effort.>> } class zone { <<zoneType>> <<Effort in training zone.>> } class percentageHeartRate { <<percentType>> <<Heart rate in percentage of maximum heart rate.>> } stopIntensity < -- percentageEffort stopIntensity < -- zone stopIntensity < -- percentageHeartRate note over stopIntensity: The intensity of the instructions. </pre>
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><xs: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	<pre> classDiagram class breath { <<xs:nonNegativeInteger>> <<Number of arm strokes per breath.>> } note over breath: Built-in derived type. The nonNegativeInteger datatype is derived from integer by setting the value of minInclusive to... </pre>
Type	xs:nonNegativeInteger
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><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></pre>

Element instructionGroup / underwater

Namespace	https://github.com/bartneck/swiML						
Annotations	True if swimming under water.						
Diagram	<pre> graph LR underwater[underwater Type xs:boolean] --> xsboolean[xs:boolean] </pre> <p>The diagram shows a UML class named "underwater" with a dependency arrow pointing to another class named "xs:boolean". A callout box below the "underwater" class contains the annotation "True if swimming under water.". A callout box below the "xs:boolean" class contains the text "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="underwater" minOccurs="0" maxOccurs="1" type="xs:boolean"> <xs:annotation> <xs:documentation>True if swimming under water.</xs:documentation> </xs:annotation> </xs:element> </pre>						

Element instructionGroup / equipment

Namespace	https://github.com/bartneck/swiML														
Annotations	Equipment to be used, such as board or pads.														
Diagram	<pre> graph LR equipment[equipment Type equipmentType] --> equipmentType[equipmentType] </pre> <p>The diagram shows a UML class named "equipment" with a dependency arrow pointing to another class named "equipmentType". A callout box below the "equipment" class contains the annotation "Equipment to be used, such as board or pads.".</p>														
Type	equipmentType														
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>unbounded</td> </tr> </table>	content:	simple	minOccurs:	0	maxOccurs:	unbounded								
content:	simple														
minOccurs:	0														
maxOccurs:	unbounded														
Facets	<table border="1"> <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														
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 instruction.				
Diagram	<pre> graph LR instructionDescription[instructionDescription Type instructionDescriptionType] --> instructionDescriptionType[instructionDescriptionType] </pre> <p>The diagram shows a UML class named "instructionDescription" with a dependency arrow pointing to another class named "instructionDescriptionType". A callout box below the "instructionDescription" class contains the annotation "Short description to modify the instruction.". A callout box below the "instructionDescriptionType" class contains the text "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> </table>	content:	simple	minOccurs:	0
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 instruction.</xs:documentation> </xs:annotation> </xs:element></pre>	

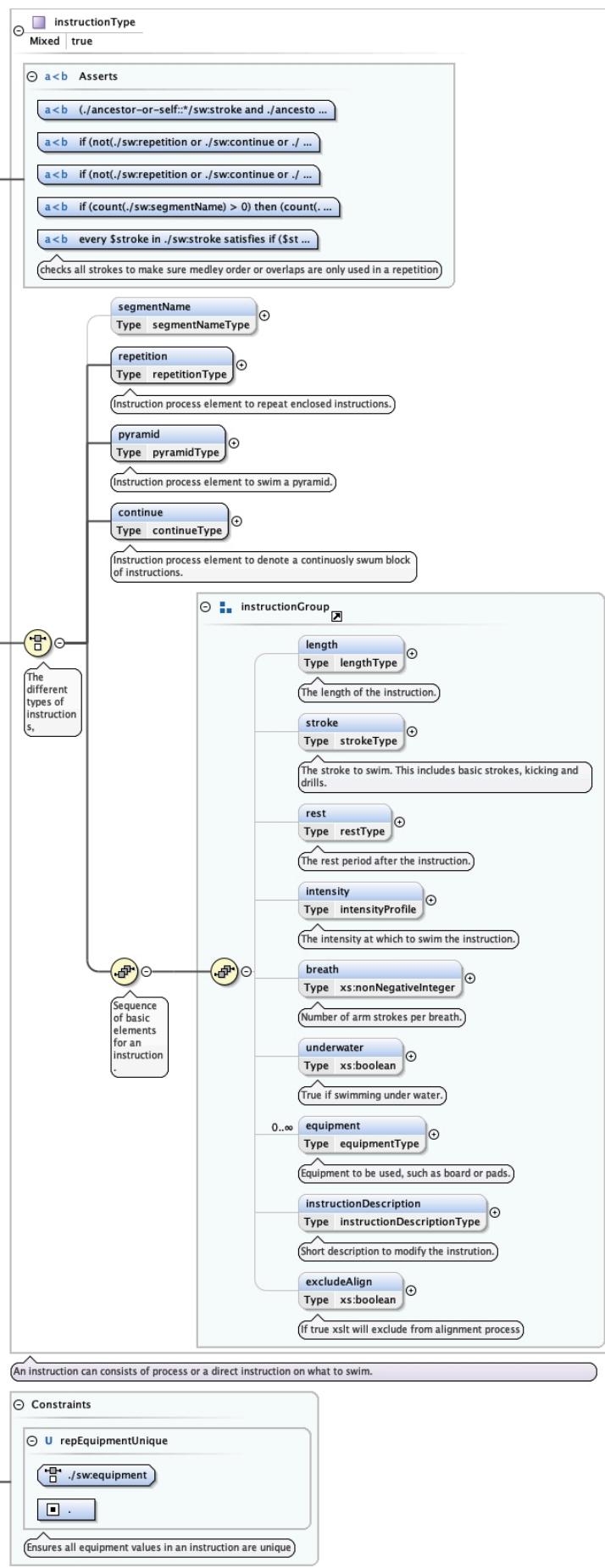
Element instructionGroup / excludeAlign

Namespace	https://github.com/bartneck/swiML						
Annotations	If true xsslt will exclude from alignment process						
Diagram	<p>The diagram shows a class named 'excludeAlign' with a multiplicity of 1. It has a directed association labeled 'Type' pointing to a class named 'xs:boolean'. A callout box points to the 'xs:boolean' class with the text: 'Built-in primitive type. It defines the boolean values true and false.' Another callout box points to the 'excludeAlign' class with the text: 'If true xsslt will exclude from alignment process'.</p>						
Type	xs:boolean						
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						
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>						

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



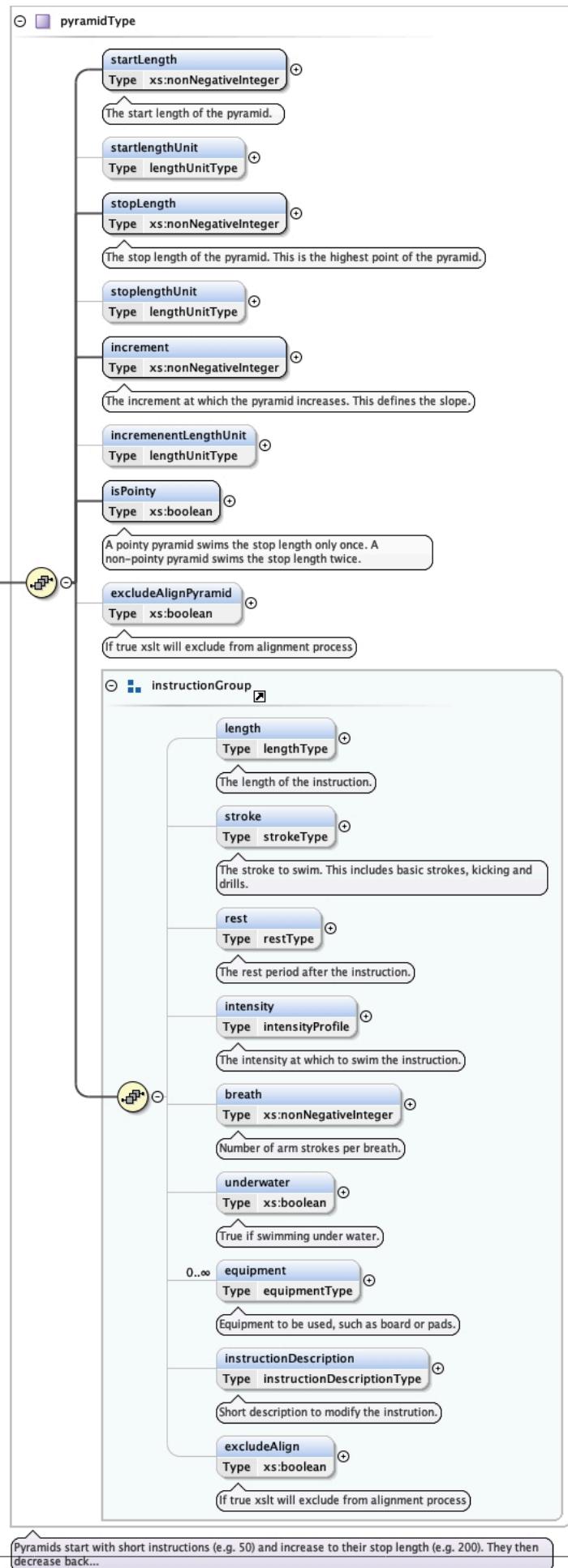
Type	instructionType											
Properties	content: complex minOccurs: 1 maxOccurs: unbounded mixed: true											
Model	segmentName{0,1} repetition pyramid continue (length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1} , excludeAlign{0,1})											
Children	breathe, 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> <p>(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName</p> <p>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())</p> <p>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())</p> <p>if (count(.//sw:segmentName) > 0) then (count(.//sw:segmentName//../*) = 0) else (true())</p> <p>every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder') then (\$stroke/parent::*/*/parent::sw:repetition) else (\$stroke/parent::*/*)</p> <p>checks all strokes to make sure medley order or overlaps are only used in a repetition</p>	XPath default namespace										
Identity constraints	<table border="1"> <thead> <tr> <th>QName</th> <th>Type</th> <th>Refer</th> <th>Selector</th> <th>Field(s)</th> </tr> </thead> <tbody> <tr> <td>repEquipmentUnique</td> <td>unique</td> <td></td> <td>/sw:equipment</td> <td>.</td> </tr> </tbody> </table>	QName	Type	Refer	Selector	Field(s)	repEquipmentUnique	unique		/sw:equipment	.	
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=".//sw:equipment"/> </xs:unique> </xs:element></pre>											

	</xs:unique> </xs:element>
--	-------------------------------

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 , startlengthUnit{0,1} , stopLength , stoplengthUnit{0,1} , increment , incremenentLengthUnit{0,1} , isPointy , excludeAlignPyramid{0,1} , 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	breadth, equipment, excludeAlign, excludeAlignPyramid, incremenentLengthUnit, increment, instructionDescription, intensity, isPointy, length, rest, startLength, startlengthUnit, stopLength, stoplengthUnit, stroke, underwater
Instance	<pre><pyramid xmlns="https://github.com/bartneck/swiML"> <startLength>{1,1}</startLength> <startlengthUnit>{0,1}</startlengthUnit> <stopLength>{1,1}</stopLength> <stoplengthUnit>{0,1}</stoplengthUnit> <increment>{1,1}</increment> <incremenentLengthUnit>{0,1}</incremenentLengthUnit> <isPointy>{1,1}</isPointy> <excludeAlignPyramids>{0,1}</excludeAlignPyramid> <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> </pyramid></pre>
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	<p>The start length of the pyramid.</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> <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><xss:element name="startLength" minOccurs="1" maxOccurs="1" type="xs:nonNegativeInteger"> <xss:annotation> <xss:documentation>The start length of the pyramid.</xss:documentation> </xss:annotation> </xss:element></pre>						

Element pyramidType / startlengthUnit

Namespace	https://github.com/bartneck/swiML						
Diagram							
Type	lengthUnitType						
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>laps</td> </tr> <tr> <td>enumeration</td> <td>yards</td> </tr> </table>	enumeration	meters	enumeration	laps	enumeration	yards
enumeration	meters						
enumeration	laps						
enumeration	yards						

	enumeration	time
Source	<xs:element name="startlengthUnit" type="lengthUnitType" minOccurs="0" maxOccurs="1"/>	

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								
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	<xs:element name="stopLength" minOccurs="1" maxOccurs="1" type="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The stop length of the pyramid. This is the highest point of the pyramid.</xs:documentation> </xs:annotation> </xs:element>							

Element pyramidType / stoplengthUnit

Namespace	https://github.com/bartneck/swiML									
Diagram										
Type	lengthUnitType									
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>laps</td> </tr> <tr> <td>enumeration</td> <td>yards</td> </tr> <tr> <td>enumeration</td> <td>time</td> </tr> </table>		enumeration	meters	enumeration	laps	enumeration	yards	enumeration	time
enumeration	meters									
enumeration	laps									
enumeration	yards									
enumeration	time									
Source	<xs:element name="stoplengthUnit" type="lengthUnitType" minOccurs="0" maxOccurs="1"/>									

Element pyramidType / increment

Namespace	https://github.com/bartneck/swiML							
Annotations	The increment at which the pyramid increases. This defines the slope.							
Diagram								
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	<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></xs:element></pre>

Element pyramidType / incremenentLengthUnit

Namespace	https://github.com/bartneck/swiML
Diagram	
Type	lengthUnitType
Properties	content: simple minOccurs: 0 maxOccurs: 1
Facets	enumeration meters enumeration laps enumeration yards enumeration time
Source	<pre><xs:element name="incremenentLengthUnit" type="lengthUnitType" 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	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 pyramidType / excludeAlignPyramid

Namespace	https://github.com/bartneck/swiML
Annotations	If true xslt will exclude from alignment process
Diagram	
Type	xs:boolean
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><xs:element name="excludeAlignPyramid" 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></pre>

Element instructionType / continue

Namespace	https://github.com/bartneck/swiML
Annotations	Instruction process element to denote a continuously swim block of instructions.
Diagram	<pre> classDiagram class continue { Type continueType "Instruction process element to denote a continuously swim block of instructions." } class assert { Type a<b "Asserts" a<b (count(././sw:instruction/sw:rest) = 0) "ensures no rest tags occur within the instructions of a continue only at top level" } class continueLength { Type xs:string "total length to swim continuously or total number of repetitions May or may not be defined but if not it will..." } class instructionGroup { length Type lengthType "The length of the instruction." stroke Type strokeType "The stroke to swim. This includes basic strokes, kicking and drills." rest Type restType "The rest period after the instruction." intensity Type intensityProfile "The intensity at which to swim the instruction." breath Type xs:nonNegativeInteger "Number of arm strokes per breath." underwater Type xs:boolean "True if swimming under water." equipment Type equipmentType "Equipment to be used, such as board or pads." instructionDescription Type instructionDescriptionType "Short description to modify the instruction." excludeAlign Type xs:boolean "If true xslt will exclude from alignment process" } class instruction { Type instructionType "A continue can include any number of instructions, which in turn can contain repetitions or any other complex..." excludeAlignContinue Type xs:boolean "If true xslt will exclude from alignment process" } </pre> <p>The diagram illustrates the structure of the <code>continue</code> element. It consists of several components:</p> <ul style="list-style-type: none"> continue: The main element, defined as <code>Type continueType</code>. Its annotation states: "Instruction process element to denote a continuously swim block of instructions." assert: An attribute of <code>continue</code>, defined as <code>Type a<b</code>. Its annotation states: "Asserts" and provides a condition: <code>a<b (count(././sw:instruction/sw:rest) = 0)</code>. A note below states: "ensures no rest tags occur within the instructions of a continue only at top level". continueLength: An attribute of <code>continue</code>, defined as <code>Type xs:string</code>. Its annotation states: "total length to swim continuously or total number of repetitions May or may not be defined but if not it will..." instructionGroup: A complex type containing the following elements: <ul style="list-style-type: none"> length: Type <code>lengthType</code>. Annotation: "The length of the instruction." stroke: Type <code>strokeType</code>. Annotation: "The stroke to swim. This includes basic strokes, kicking and drills." rest: Type <code>restType</code>. Annotation: "The rest period after the instruction." intensity: Type <code>intensityProfile</code>. Annotation: "The intensity at which to swim the instruction." breath: Type <code>xs:nonNegativeInteger</code>. Annotation: "Number of arm strokes per breath." underwater: Type <code>xs:boolean</code>. Annotation: "True if swimming under water." equipment: Type <code>equipmentType</code>. Annotation: "Equipment to be used, such as board or pads." instructionDescription: Type <code>instructionDescriptionType</code>. Annotation: "Short description to modify the instruction." excludeAlign: Type <code>xs:boolean</code>. Annotation: "If true xslt will exclude from alignment process" instruction: A sequence of zero or more <code>instruction</code> elements, defined as <code>1..> instruction</code> with <code>Type instructionType</code>. Its annotation states: "A continue can include any number of instructions, which in turn can contain repetitions or any other complex...". excludeAlignContinue: An attribute of <code>instruction</code>, defined as <code>Type xs:boolean</code>. Its annotation states: "If true xslt will exclude from alignment process". <p>A general note at the bottom states: "Continues for when different strokes, equipment or styles need to be swum continuously".</p>
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} , excludeAlign{0,1} , instruction+ , excludeAlignContinue{0,1}	
Children	breath, continueLength, equipment, excludeAlign, excludeAlignContinue, 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> <equipments>{0,unbounded}</equipment> <instructionDescription>{0,1}</instructionDescription> <excludeAlign>{0,1}</excludeAlign> <instruction>{1,unbounded}</instruction> <excludeAlignContinue>{0,1}</excludeAlignContinue> </continue></pre>	
Asserts	Test (count(.//sw:instruction/sw:rest) = 0)	XPath default namespace
	ensures no rest tags occur within the instructions of a continue only at top level	
Source	<pre><xss:element name="continue" type="continueType"> <xss:annotation> <xss:documentation>Instruction process element to denote a continuously swum block of instructions.</xss:documentation> </xss:annotation> </xss:element></pre>	

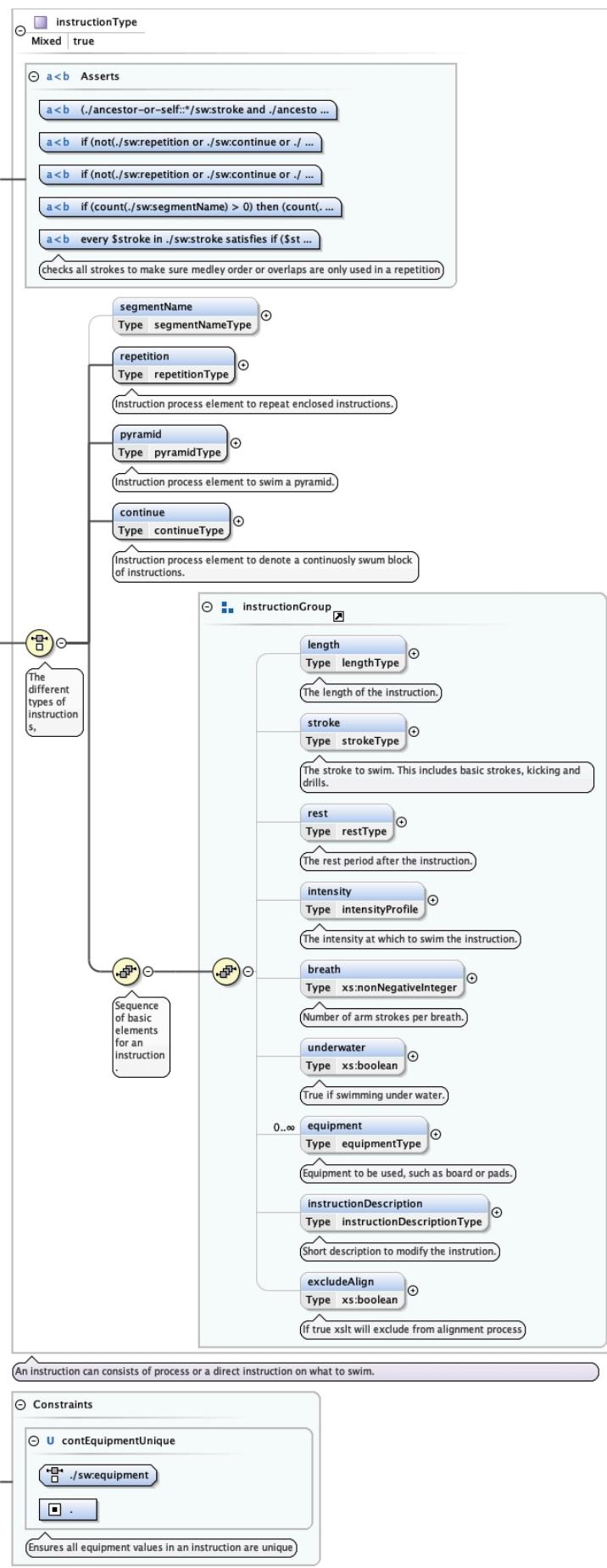
Element continueType / continueLength

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	<p>The diagram shows a UML class diagram fragment. A rounded rectangle labeled "continueLength" has an association line pointing to another rounded rectangle labeled "xs:string". A callout box points to the "xs:string" box with the text: "total length to swim continuously or total number of repetitions May or may not be defined but if not it will...". Another callout box points to the "xs:string" box with the text: "Built-in primitive type. The string datatype represents character strings in XML."</p>
Type	xs:string
Properties	content: simple minOccurs: 0 maxOccurs: 1
Source	<pre><xss:element name="continueLength" minOccurs="0" maxOccurs="1" type="xs:string"> <xss:annotation> <xss: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</xss:documentation> </xss:annotation> </xss: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	<table border="1"> <thead> <tr> <th>Test</th> <th>XPath default namespace</th> </tr> </thead> <tbody> <tr> <td>(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName</td><td></td></tr> <tr> <td>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())</td><td></td></tr> <tr> <td>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())</td><td></td></tr> <tr> <td>if (count(.//sw:segmentName) > 0) then (count(.//sw:segmentName//../*) = 0) else (true())</td><td></td></tr> <tr> <td>every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder') then (\$stroke/parent::*//parent:sw:repetition) else (\$stroke/parent::*//parent:sw:repetition)</td><td></td></tr> <tr> <td>checks all strokes to make sure medley order or overlaps are only used in a repetition</td><td></td></tr> </tbody> </table>		Test	XPath default namespace	(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName		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())		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())		if (count(.//sw:segmentName) > 0) then (count(.//sw:segmentName//../*) = 0) else (true())		every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder') then (\$stroke/parent::*//parent:sw:repetition) else (\$stroke/parent::*//parent:sw:repetition)		checks all strokes to make sure medley order or overlaps are only used in a repetition	
Test	XPath default namespace															
(./ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName																
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())																
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())																
if (count(.//sw:segmentName) > 0) then (count(.//sw:segmentName//../*) = 0) else (true())																
every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder') then (\$stroke/parent::*//parent:sw:repetition) else (\$stroke/parent::*//parent:sw:repetition)																
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=".//sw:equipment"/> </xs:unique> </xs:element></pre>															

```
</xs:unique>
</xs:element>
```

Element continueType / excludeAlignContinue

Namespace	https://github.com/bartneck/swiML						
Annotations	If true xslt will exclude from alignment process						
Diagram	<p>The diagram shows the element <code>excludeAlignContinue</code> with its type <code>xs:boolean</code>. A callout box points to the type with the text "If true xslt will exclude from alignment process". Another callout box points to the <code>xs:boolean</code> type with the text "Built-in primitive type. It defines the boolean values true and false.".</p>						
Type	<code>xs:boolean</code>						
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						
Source	<pre><xs:element name="excludeAlignContinue" 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></pre>						

Element repetitionType / excludeAlignRepetition

Namespace	https://github.com/bartneck/swiML						
Annotations	If true xslt will exclude from alignment process						
Diagram	<p>The diagram shows the element <code>excludeAlignRepetition</code> with its type <code>xs:boolean</code>. A callout box points to the type with the text "If true xslt will exclude from alignment process". Another callout box points to the <code>xs:boolean</code> type with the text "Built-in primitive type. It defines the boolean values true and false.".</p>						
Type	<code>xs:boolean</code>						
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						
Source	<pre><xs:element name="excludeAlignRepetition" 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></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 the simple type <code>titleString</code> as a restriction of <code>xs:string</code>. A callout box points to <code>titleString</code> with the text "The length of the title is constraint in length.". Another callout box points to <code>xs:string</code> 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 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:simpleType></pre>

```

<xs:restriction base="xs:string">
  <xsmaxLength value="60"/>
</xs:restriction>
</xs:simpleType>

```

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	<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>	

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	<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>	

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	enumeration meters enumeration yards	
Used by	Element	program/lengthUnit
Source	<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="yards"/>
</xs:restriction>
</xs:simpleType>

```

Simple Type segmentNameType

Namespace	https://github.com/bartneck/swiML	
Diagram	<p>The diagram shows a class named "segmentNameType" with a hollow circle symbol indicating it is a restriction of the "xs:string" type. A callout box below the association line states: "Built-in primitive type. The string datatype represents character strings in XML."</p>	
Type	restriction of xs:string	
Facets	maxLength 30	
Used by	Element instructionType/segmentName	
Source	<pre> <xs:simpleType name="segmentNameType"> <xs:restriction base="xs:string"> <xs:maxLength value=" 30 "/> </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>The diagram shows a class named "instructionDescriptionType" with a hollow circle symbol indicating it is a restriction of the "xs:string" type. Two callout boxes below the association line state: "The length of the description text is constraint in length." and "Built-in primitive type. The string datatype represents character strings in XML."</p>	
Type	restriction of xs:string	
Facets	maxLength 100	
Used by	Elements instructionGroup/instructionDescription, repetitionType/repetitionDescription	
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>	

Simple Type standardStrokeType

Namespace	https://github.com/bartneck/swiML													
Diagram	<p>The diagram shows a class named "standardStrokeType" with a hollow circle symbol indicating it is a restriction of the "xs:string" type. A callout box below the association line states: "Built-in primitive type. The string datatype represents character strings in XML."</p>													
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> </table>		enumeration	butterfly	enumeration	backstroke	enumeration	breaststroke	enumeration	freestyle	enumeration	individualMedley	enumeration	reverseIndividualMedley
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	<p>Built-in primitive type. The string datatype represents character strings in XML.</p>															
Type	restriction of xs:string															
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															
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>															

<pre></xs:simpleType></pre>

Simple Type legMovementType

Namespace	https://github.com/bartneck/swiML							
Diagram	<p>The diagram shows a class named 'legMovementType' with a hollow circle indicating it is a restriction of the 'xs:string' primitive type. Three arrows point from the string type to the class, each labeled with an enumeration value: 'flutter', 'dolpine', and 'scissor'. A callout box points to the 'xs:string' type with the text: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>							
Type	restriction of xs:string							
Facets	<table border="1"> <tr> <td>enumeration</td> <td>flutter</td> </tr> <tr> <td>enumeration</td> <td>dolpine</td> </tr> <tr> <td>enumeration</td> <td>scissor</td> </tr> </table>		enumeration	flutter	enumeration	dolpine	enumeration	scissor
enumeration	flutter							
enumeration	dolpine							
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="dolpine"/> <xs:enumeration value="scissor"/> </xs:restriction> </xs:simpleType></pre>							

Simple Type drillNameType

Namespace	https://github.com/bartneck/swiML																																			
Annotations	Drill names.																																			
Diagram	<p>The diagram shows a class named 'drillNameType' with a hollow circle indicating it is a restriction of the 'xs:string' primitive type. An arrow points from the string type to the class, labeled 'Drill names.'. A callout box points to the 'xs:string' type with the text: 'Built-in primitive type. The string datatype represents character strings in XML.'</p>																																			
Type	restriction of xs:string																																			
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>fingerDrag</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>6666</td> </tr> <tr> <td>enumeration</td> <td>3Kick1Pull</td> </tr> <tr> <td>enumeration</td> <td>other</td> </tr> </table>		enumeration	6KickDrill	enumeration	8KickDrill	enumeration	10KickDrill	enumeration	12KickDrill	enumeration	fingerTrails	enumeration	fingerDrag	enumeration	123	enumeration	bigDog	enumeration	scull	enumeration	singleArm	enumeration	any	enumeration	technic	enumeration	dogPaddle	enumeration	tarzan	enumeration	6666	enumeration	3Kick1Pull	enumeration	other
enumeration	6KickDrill																																			
enumeration	8KickDrill																																			
enumeration	10KickDrill																																			
enumeration	12KickDrill																																			
enumeration	fingerTrails																																			
enumeration	fingerDrag																																			
enumeration	123																																			
enumeration	bigDog																																			
enumeration	scull																																			
enumeration	singleArm																																			
enumeration	any																																			
enumeration	technic																																			
enumeration	dogPaddle																																			
enumeration	tarzan																																			
enumeration	6666																																			
enumeration	3Kick1Pull																																			
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"/></pre>																																			

```

<xs:enumeration value="12KickDrill"/>
<xs:enumeration value="fingerTrails"/>
<xs:enumeration value="fingerDrag"/>
<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="6666"/>
<xs:enumeration value="3Kick1Pull"/>
<xs:enumeration value="other"/>
</xs:restriction>
</xs:simpleType>

```

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> <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 intensity zone.</p> <p>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>										

<pre></xs:simpleType></pre>

Simple Type equipmentType

Namespace	https://github.com/bartneck/swiML	
Diagram		
Type	restriction of xs:string	
Facets	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 lengthUnitType

Namespace	https://github.com/bartneck/swiML	
Diagram		
Type	restriction of xs:string	
Facets	enumeration meters enumeration laps enumeration yards enumeration time	
Used by	Elements	pyramidType/incremenentLengthUnit, pyramidType/startlengthUnit, pyramidType/stoplenthUnit
Source	<pre><xs:simpleType name="lengthUnitType"> <xs:restriction base="xs:string"> <xs:enumeration value="meters"/> <xs:enumeration value="laps"/> <xs:enumeration value="yards"/> <xs:enumeration value="time"/> </xs:restriction> </xs:simpleType></pre>	

Simple Type equipmentList

Namespace	https://github.com/bartneck/swiML	
Diagram		
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	<pre> classDiagram class instructionType { <<Mixed true>> <<An instruction can consists of process or a direct instruction on what to swim.>> } class segmentName { Type segmentNameType } class repetition { Type repetitionType } class pyramid { Type pyramidType } class continue { Type continueType } class length { Type lengthType } class stroke { Type strokeType } class rest { Type restType } class intensity { Type intensityProfile } class breath { Type xs:nonNegativeInteger } class underwater { Type xs:boolean } class equipment { Type equipmentType } class instructionDescription { Type instructionDescriptionType } class excludeAlign { Type xs:boolean } instructionType --> segmentName instructionType --> repetition instructionType --> pyramid instructionType --> continue instructionType --> length instructionType --> stroke instructionType --> rest instructionType --> intensity instructionType --> breath instructionType --> underwater instructionType --> equipment instructionType --> instructionDescription instructionType --> excludeAlign </pre> <p>The diagram illustrates the structure of the <code>instructionType</code> complex type. It is defined as a mixed type, meaning it can contain both element declarations and text content. The type is annotated with the text: "An instruction can consists of process or a direct instruction on what to swim."</p> <p>The <code>instructionType</code> is composed of several sub-elements:</p> <ul style="list-style-type: none"> <code>segmentName</code>: Type <code>segmentNameType</code>. Description: "Instruction process element to repeat enclosed instructions." <code>repetition</code>: Type <code>repetitionType</code>. Description: "Instruction process element to repeat enclosed instructions." <code>pyramid</code>: Type <code>pyramidType</code>. Description: "Instruction process element to swim a pyramid." <code>continue</code>: Type <code>continueType</code>. Description: "Instruction process element to denote a continuously swum block of instructions." <code>length</code>: Type <code>lengthType</code>. Description: "The length of the instruction." <code>stroke</code>: Type <code>strokeType</code>. Description: "The stroke to swim. This includes basic strokes, kicking and drills." <code>rest</code>: Type <code>restType</code>. Description: "The rest period after the instruction." <code>intensity</code>: Type <code>intensityProfile</code>. Description: "The intensity at which to swim the instruction." <code>breath</code>: Type <code>xs:nonNegativeInteger</code>. Description: "Number of arm strokes per breath." <code>underwater</code>: Type <code>xs:boolean</code>. Description: "True if swimming under water." <code>equipment</code>: Type <code>equipmentType</code>. Description: "Equipment to be used, such as board or pads." <code>instructionDescription</code>: Type <code>instructionDescriptionType</code>. Description: "Short description to modify the instruction." <code>excludeAlign</code>: Type <code>xs:boolean</code>. Description: "If true xslt will exclude from alignment process" <p>A separate section titled "Sequence of basic elements for an instruction" shows the sequence of elements: <code>instructionType</code>, <code>segmentName</code>, <code>repetition</code>, <code>pyramid</code>, <code>continue</code>, <code>length</code>, <code>stroke</code>, <code>rest</code>, <code>intensity</code>, <code>breath</code>, <code>underwater</code>, <code>equipment</code>, <code>instructionDescription</code>, and <code>excludeAlign</code>.</p>
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		breathe, continue, equipment, excludeAlign, instructionDescription, intensity, length, pyramid, repetition, rest, segmentName, stroke, underwater
Asserts	Test (/ancestor-or-self::*/sw:stroke and ./ancestor-or-self::*/sw:length) or ./sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName 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()) 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()) if(count(.//sw:segmentName) > 0) then (count(.//sw:segmentName/../*) = 0) else (true()) every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder') then (\$stroke/parent::*//parent:sw:repetition) else (\$stroke/parent::*)	XPath default namespace 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()) 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()) if(count(.//sw:segmentName) > 0) then (count(.//sw:segmentName/../*) = 0) else (true()) every \$stroke in ./sw:stroke satisfies if (\$stroke/sw:standardStroke = 'individualMedleyOverlap' or \$stroke/sw:standardStroke = 'individualMedleyOrder' or \$stroke/sw:standardStroke = 'reverseIndividualMedleyOrder') then (\$stroke/parent::*//parent:sw:repetition) else (\$stroke/parent::*)
Source	<pre> <xs:complexType name="instructionType" mixed="true"> <xs:annotation> <xs:documentation>An instruction can consists of process or a direct instruction on what to swim.</xs:documentation> </xs:annotation> <xs:choice> <xs:annotation> <xs:documentation>The different types of instructions,</xs:documentation> </xs:annotation> <!-- ===== --> <!-- Process based elements for instructions --> <xs:element name="segmentName" minOccurs="0" maxOccurs="1" type="segmentNameType" /> <xs:element name="repetition" type="repetitionType" /> <xs:annotation> <xs:documentation>Instruction process element to repeat enclosed instructions.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="pyramid" type="pyramidType" /> <xs:annotation> <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:sequence> <xs:annotation> <xs:documentation>Sequence of basic elements for an instruction.</xs:documentation> </xs:annotation> <!-- Common elements for instructions --> <xs:group ref="instructionGroup" /> </pre>	

```

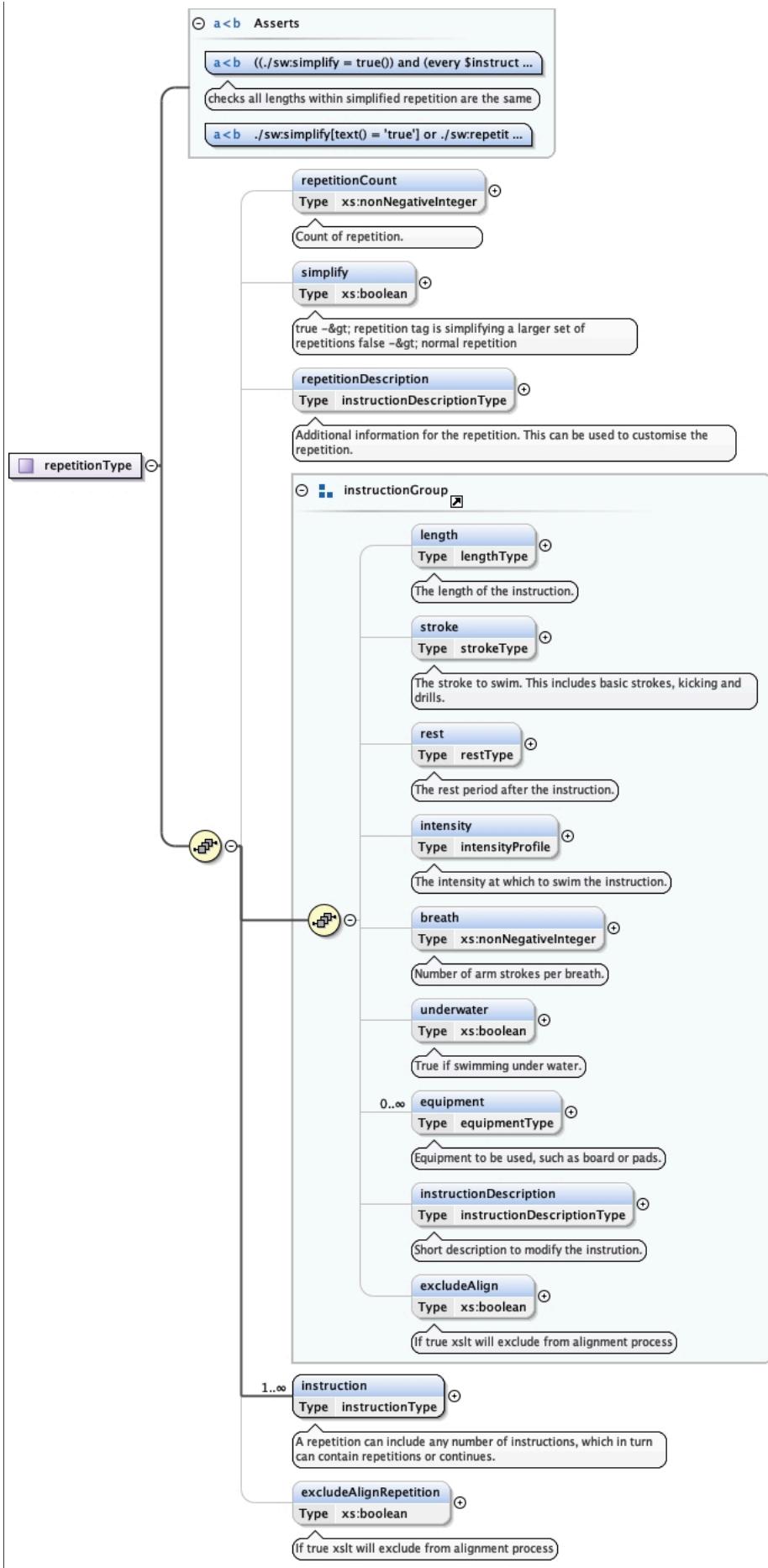
        </xs:sequence>
    </xs:choice>
    <!-- checks every instruction has stroke, rest and length defined
        any other element in an instruction doesnt have to be defined-->
    <xs:assert test="                               (.//ancestor-or-self::*/sw:stroke
                                                and .//ancestor-
or-self::*/*sw:length)           or ./sw:repetition          or ./sw:continue
                                or ./sw:pyramid       or ./sw:segmentName")/>
    <!-- checks every instruction doesnt have repetitions of elements defined and cannot be extended
-->
    <xs:assert test="                     if (not(.//sw:repetition
                                                or ./sw:segmentName))
                                (               every $element in ./*
                                    every $match in ./ancestor::*[name() = 'instruction' or name() = 'repetition' or name()
= 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'length' or name() = 'stroke' or name() = 'rest' or name() = 'intensity'
or name() = 'breath' or name() = 'underwater']
                                    not(name($element) = name($match))           satisfies
                                (true())                                         ))           else
                                (true())                                         "/>
    <xs:assert test="                     if (not(.//sw:repetition
                                                or ./sw:segmentName))
                                (               every $element in ./*
                                    every $match in ./ancestor::*[name() = 'instruction' or name()
= 'repetition' or name() = 'continue' or name() = 'pyramid'][not(.//sw:repetition or ./sw:continue
or ./sw:pyramid or ./sw:segmentName)]/*[name() = 'equipment']
                                    not($element/text() = $match/text())           satisfies
                                (true())                                         ))
                                (true())                                         "/>
    <!--checks all segment names are only at top level-->
    <xs:assert test="                     if (count(.//sw:segmentName) > 0)
                                                then
                                (count(.//sw:segmentName//...//ancestor::*)) = 0)           else
                                (true())                                         "/>
    <!-- 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')
                                then
                                (parent/*/parent::sw:repetition)           else
                                (parent/*/parent::sw:repetition)
                                (true())                                         ">
        <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} , excludeAlign{0,1} , instruction+ , excludeAlignRepetition{0,1}	
Children	breath, equipment, excludeAlign, excludeAlignRepetition, instruction, instructionDescription, intensity, length, repetitionCount, repetitionDescription, rest, simplify, stroke, underwater	
Asserts	<p>Test</p> <pre>((./sw:simplify = true()) and (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)][not(.//sw:continue/sw:continueLength)]) then(if(count(\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(.//sw:continue/sw:continueLength)][1])//sw:lengthAsDistance) else(sum((\$instruction/descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(.//sw:continue/sw:continueLength)][1])//sw:lengthAsDistance))) else(0) + (if(\$instruction/descendant-or-self::sw:continueLength) else(0)) = number(((./descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(.//sw:continue/sw:continueLength)][1])//sw:lengthAsDistance) ./descendant-or-self::sw:continueLength)[1])) or ./sw:simplify = false() or count(.//sw:simplify) = 0 checks all lengths within simplified repetition are the same ./sw:simplify[text() = 'true'] or ./sw:repetitionCount and not(.//sw:simplify[text() = 'true'] and ./sw:repetitionCount)</pre>	XPath default namespace
Source	<pre><xs:complexType name="repetitionType"> <xs:annotation> <xs:documentation></pre>	

```

<xs:annotation>
    <xs:documentation>If true xslt will exclude from alignment process</xs:documentation>
    </xs:annotation>
</xs:element>
</xs:sequence>
<!-- ===== -->
<!-- Assertions -->
<xs:assert test="(./sw:simplify = true()) and (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)][not(./
sw:continue/sw:continueLength)]) then(if(count($instruction/
descendant-or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(./
sw:continue/sw:continueLength)]) = 1) then(number($instruction/descendant-
or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(./
sw:continue/sw:continueLength)][1])//sw:lengthAsDistance) else(sum($instruction/descendant-
or-self::sw:instruction[not(ancestor::sw:continue/sw:continueLength)][not(./
sw:continue/sw:continueLength)][1])//sw:lengthAsDistance)) ) else(0)
        + (if($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)][not(./
sw:continue/sw:continueLength)][1])//sw:lengthAsDistance) | ./.descendant-or-
self::sw:continueLength[1])) ) or ./sw:simplify = false() or count(.//sw:simplify) = 0">
<xs:annotation>
    <xs:documentation>checks all lengths within simplified repetition are the same</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.>> } lengthType < -- lengthAsDistance : "lengthAsDistance Type xs:nonNegativeInteger Length of instruction as distance." lengthType < -- lengthAsTime : "lengthAsTime Type xs:duration Duration starts with PT followed by int M and int S. For example PT1M30S for 1:30." lengthType < -- lengthAsLaps : "lengthAsLaps Type xs:nonNegativeInteger Length of instruction in number of laps." note over lengthType: Length can be described as distance or time. </pre>
Properties	mixed: true
Used by	Element instructionGroup/length
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:choice> </xs:complexType> </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>
</xs:choice>
</xs:complexType>

```

Complex Type strokeType

Namespace	https://github.com/bartneck/swiML
Annotations	Stroke types.
Diagram	<pre> graph LR strokeType["strokeType Mixed true"] --> standardStroke["standardStroke Type standardStrokeType"] strokeType --> kicking["kicking Type kickStyle"] strokeType --> drill["drill Type drillType"] </pre> <p>The diagram illustrates the structure of the <code>strokeType</code> complex type. It is defined as a mixed element, indicated by the <code>Mixed true</code> annotation. Three children are defined under this type: <code>standardStroke</code>, <code>kicking</code>, and <code>drill</code>. Each child is associated with its respective type: <code>standardStrokeType</code>, <code>kickStyle</code>, and <code>drillType</code>.</p>
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	<pre> graph LR kickStyle["kickStyle"] --> orientation["orientation Type orientationType"] kickStyle --> legMovement["legMovement Type legMovementType"] kickStyle --> standardKick["standardKick Type standardStrokeType"] </pre> <p>The diagram illustrates the structure of the <code>kickStyle</code> complex type. It is defined as a choice element, indicated by the UML stereotype <code>choice</code>. Three children are defined under this type: <code>orientation</code>, <code>legMovement</code>, and <code>standardKick</code>. Each child is associated with its respective type: <code>orientationType</code>, <code>legMovementType</code>, and <code>standardStrokeType</code>. Annotations provide descriptions for each child element.</p>
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>

```
</xs:choice>
</xs:complexType>
```

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

Complex Type restType

Namespace	https://github.com/bartneck/swiML
Annotations	The length units for a rest after a swimming instruction.
Diagram	<pre> classDiagram class restType { afterStop sinceStart sinceLastRest inOut } afterStop < -- xs:duration sinceStart < -- xs:duration sinceLastRest < -- xs:duration inOut < -- xs:nonNegativeInteger </pre>
Properties	mixed: true
Used by	Element instructionGroup/rest
Model	afterStop sinceStart sinceLastRest inOut
Children	afterStop, inOut, sinceLastRest, sinceStart
Source	<pre> <xs:complexType name="restType" mixed="true"> <xs:annotation> </pre>

```

<xs:documentation>The length units for a rest after a swimming instruction.</xs:documentation>
</xs:annotation>
<xs:choice>
  <xs:element name="afterStop" type="xs:duration">
    <xs:annotation>
      <xs: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.</xs:documentation>
    </xs:annotation>
  </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>

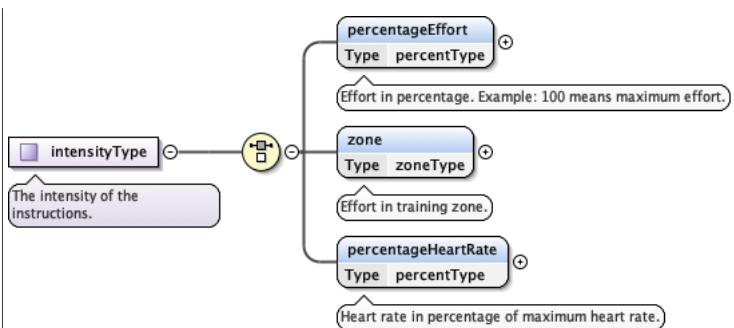
```

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 } class startIntensity { Type intensityType } class stopIntensity { Type intensityType } intensityProfile "1" --> "1" startIntensity intensityProfile "1" --> "1" stopIntensity note left of startIntensity: 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

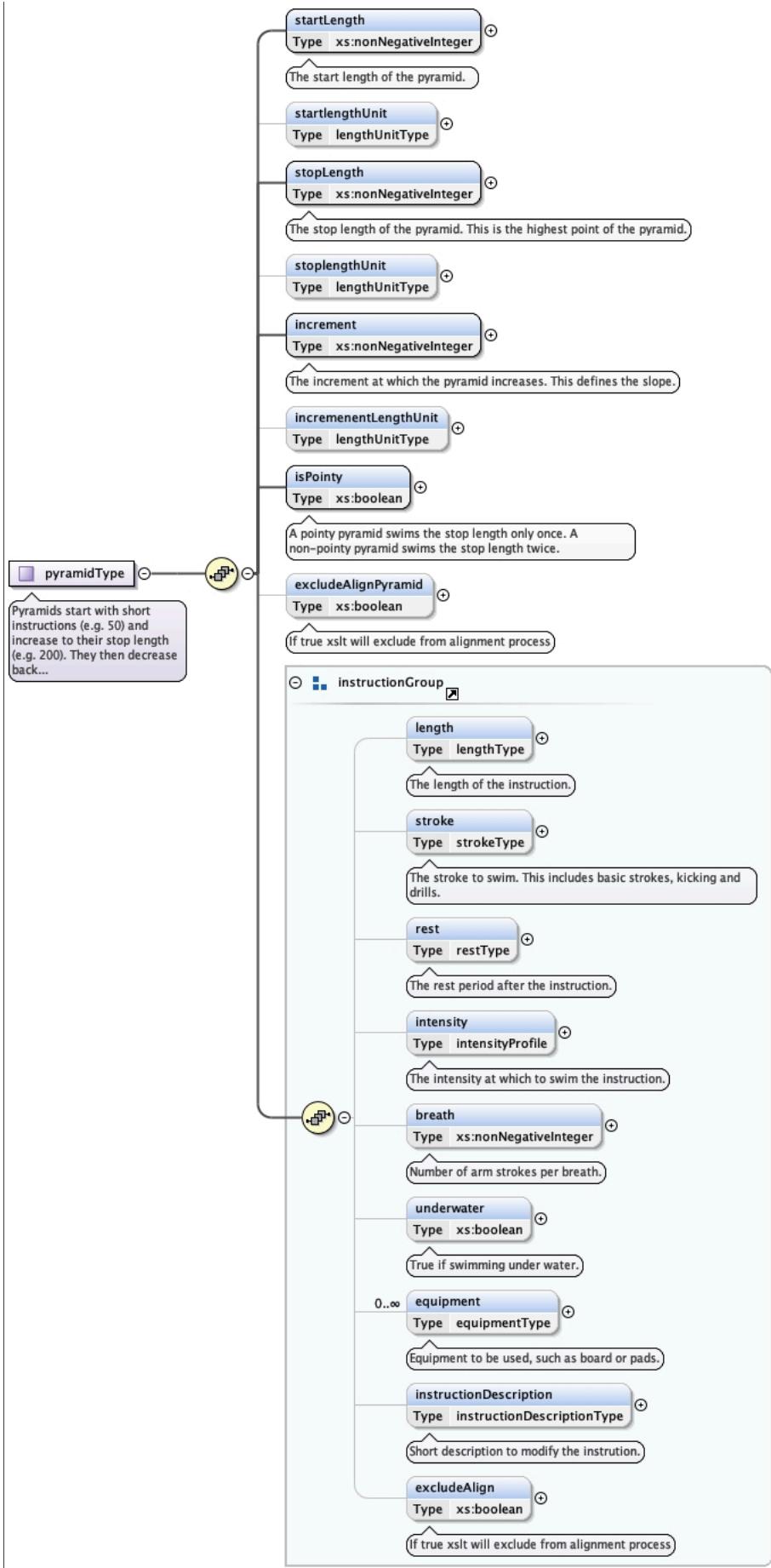
Namespace	https://github.com/bartneck/swiML
Annotations	The intensity of the instructions.

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

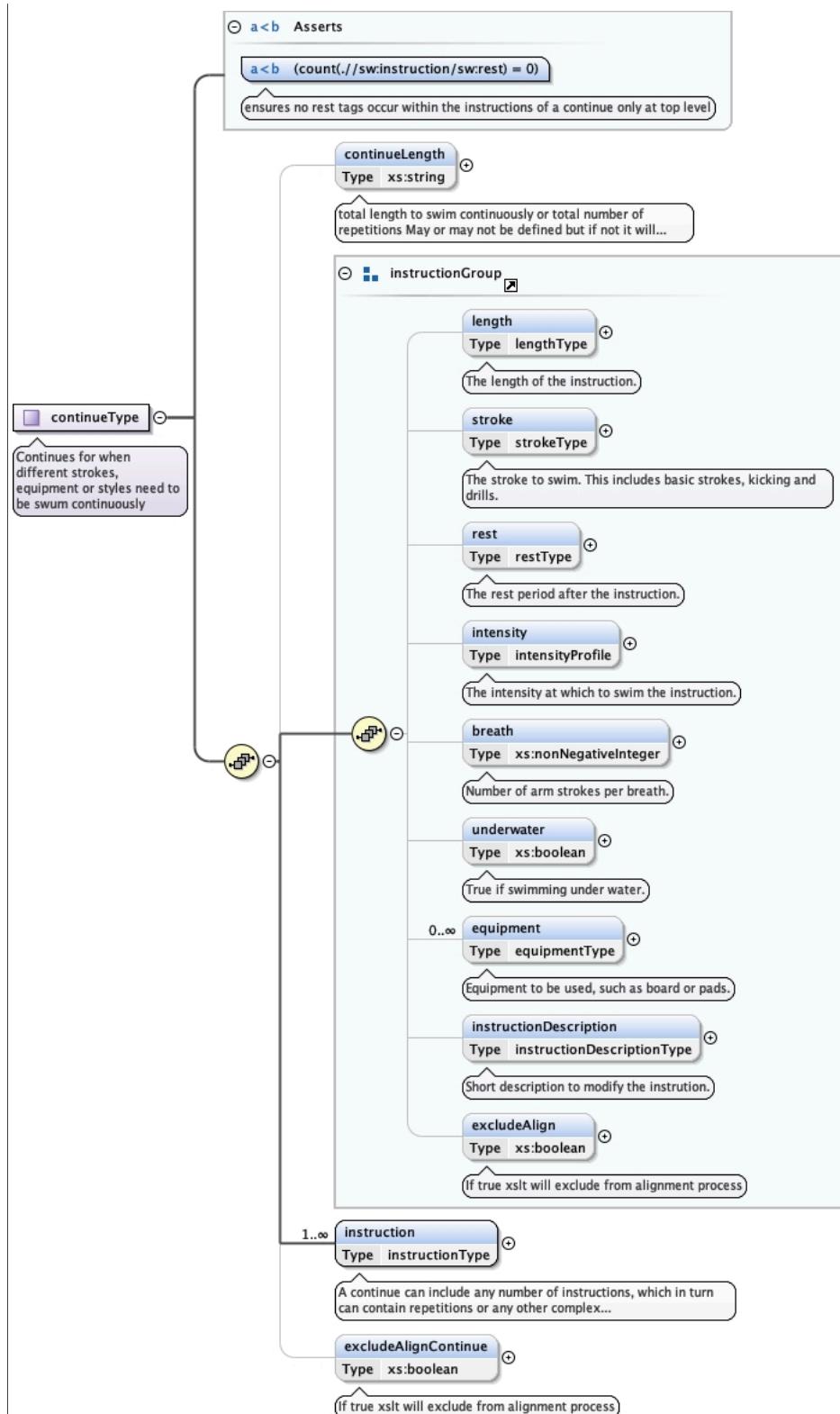


Used by	Element instructionType/pyramid
Model	startLength , startlengthUnit{0,1} , stopLength , stoplengthUnit{0,1} , increment , incremenentLengthUnit{0,1} , isPointy , excludeAlignPyramid{0,1} , length{0,1} , stroke{0,1} , rest{0,1} , intensity{0,1} , breath{0,1} , underwater{0,1} , equipment* , instructionDescription{0,1} , excludeAlign{0,1}
Children	breath, equipment, excludeAlign, excludeAlignPyramid, incremenentLengthUnit, increment, instructionDescription, intensity, isPointy, length, rest, startLength, startlengthUnit, stopLength, stoplengthUnit, stroke, underwater
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="xs:nonNegativeInteger"> <xs:annotation> <xs:documentation>The start length of the pyramid.</xs:documentation> </xs:annotation> </xs:element> <xs:element name="startlengthUnit" type="lengthUnitType" minOccurs="0" maxOccurs="1"/> <xs:element name="stopLength" minOccurs="1" maxOccurs="1" type="xs:nonNegativeInteger"> <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="stoplengthUnit" type="lengthUnitType" minOccurs="0" maxOccurs="1"/> <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="incremenentLengthUnit" type="lengthUnitType" 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:element name="excludeAlignPyramid" 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:group ref="instructionGroup"/> </xs:sequence> </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} , excludeAlign{0,1} , instruction+ , excludeAlignContinue{0,1}	
Children	breathe, continueLength, equipment, excludeAlign, excludeAlignContinue, instruction, instructionDescription, intensity, length, rest, stroke, underwater	
Asserts	Test (count(.//sw:instruction/sw:rest) = 0)	XPath default namespace

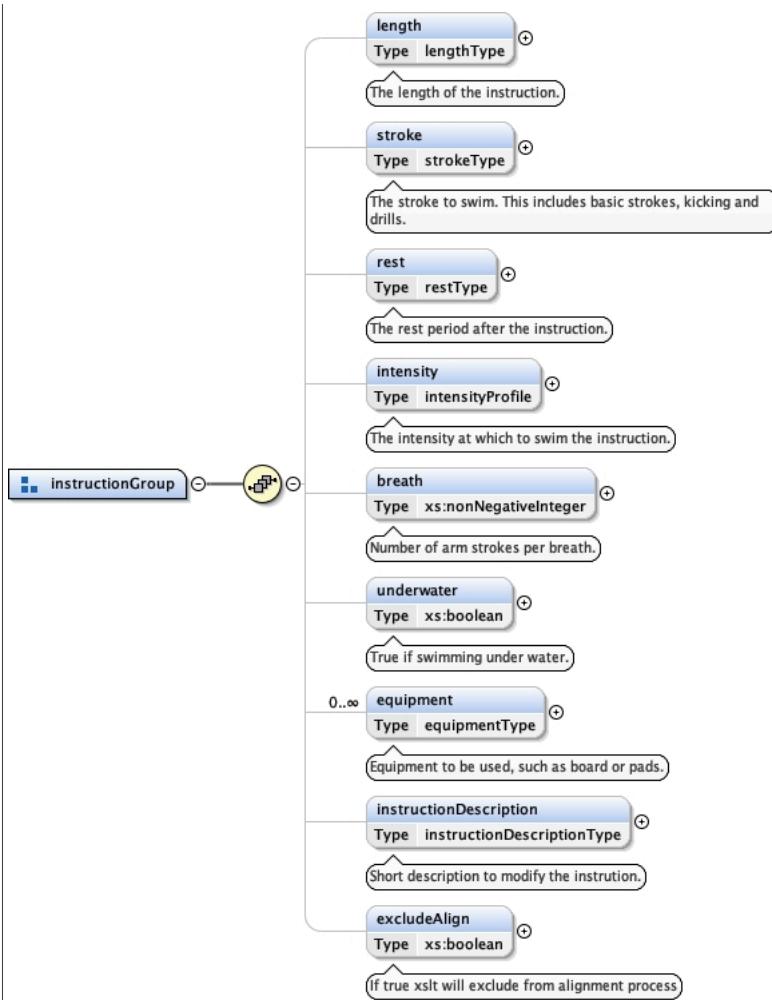
	Test	XPath default namespace
	ensures no rest tags occur within the instructions of a continue only at top level	
Source	<pre> <xs:complexType name="continueType"> <xs:annotation> <xs:documentation>Continues for when different strokes, equipment or styles need to be swum continuously</xs:documentation> </xs:annotation> <xs:sequence> <xs:element name="continueLength" minOccurs="0" maxOccurs="1" type="xs:string"> <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=".//sw:equipment"/> </xs:unique> </xs:element> <xs:element name="excludeAlignContinue" 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> <!-- ===== --> <!-- 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></pre>	

Element Group(s)

Element Group instructionGroup

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

Diagram



Used by	Complex Types continueType, instructionType, pyramidType, repetitionType
Model	<code>length{0,1}</code> , <code>stroke{0,1}</code> , <code>rest{0,1}</code> , <code>intensity{0,1}</code> , <code>breath{0,1}</code> , <code>underwater{0,1}</code> , <code>equipment*</code> , <code>instructionDescription{0,1}</code> , <code>excludeAlign{0,1}</code>
Children	breath, equipment, excludeAlign, instructionDescription, intensity, length, rest, stroke, underwater
Source	<pre> <xs:group name="instructionGroup"> <xs:sequence> <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> </pre>

```
<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: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>
</xs:group>
```