
Meta Resource Management System

Static Model

René Freude (707168) <rene.freude@gmx.de>
Daniel Sadilek (707297) <sadilek@tfh-berlin.de>
Stephan Weiß (706830) <steph.weiss@web.de>

Copyright © 2003

2003-06-15

Versionsgeschichte

Version 0.1

2003-05-19

Initial public release.

Version 0.2

2003-06-09

Corrected some cardinalities, extended descriptions, added operations.

Version 0.3

2003-06-15

Added "user logs in" sequence diagram.

This document contains the class diagrams and class descriptions that resulted from the static analysis.

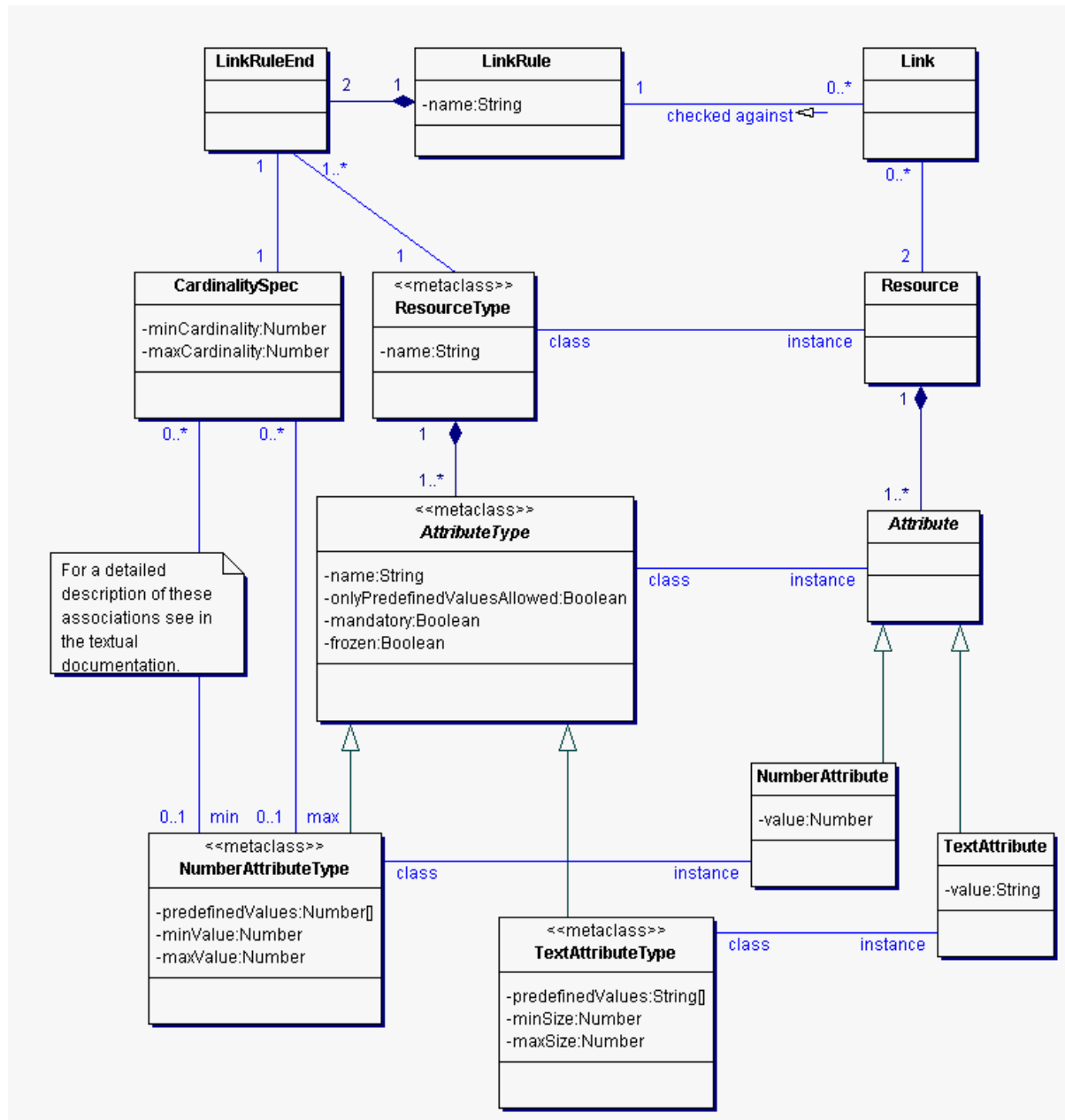
Inhaltsverzeichnis

1. Data and Meta Data Management Classes	2
1.1. Resource	2
1.2. ResourceType	3
1.3. AttributeType	3
1.4. NumberAttributeType	3
1.5. TextAttributeType	3
1.6. Attribute	3
1.7. NumberAttribute	4
1.8. TextAttribute	4
1.9. Link	4
1.10. LinkRule	4
1.11. LinkRuleEnd	4
1.12. CardinalitySpec	5
2. User Management Classes	5
2.1. AuthenticationData	6
2.2. User	6
2.3. Role	7
2.4. AccessRight	7
2.5. ResourceAccessRight	8
2.6. AttributeAccessRight	8
2.7. LinkAccessRight	8
3. Filter Classes	8
3.1. Filter	9
3.2. Constraint	10
3.3. AttributeConstraint	10
3.4. LinkageConstraint	11
4. Control and Boundary Classes	11
4.1. SessionState	11
4.2. UI	12
4.3. Sequence: User logs in	15
5. Extensive Overview	16

1. Data and Meta Data Management Classes

The following diagram depicts the data and meta data structures of the MRMS.

Abbildung 1. Data and Meta Data Classes



1.1. Resource

Description A *Resource* is composed of its *Attributes* and is an instance of a *ResourceType* which specifies which *Attributes* the *Resource* may have.

Attributes ---

Operations ---

1.2. ResourceType

Description	A <i>ResourceType</i> specifies (by composition) the <i>Attributes</i> that a <i>Resource</i> of this type has.
Attributes	<i>name</i> (String): the name of the <i>ResourceType</i>
Operations	---

1.3. AttributeType

Description	Abstract base class for attribute types that a <i>ResourceType</i> is composed of.
Attributes	<i>name</i> (String): the name of the <i>AttributeType</i> <i>onlyPredefinedValuesAllowed</i> (Boolean): if true, the user may only select the predefined values for an <i>Attribute</i> that has this type; if false, he may enter another value as well <i>mandatory</i> (Boolean): if true, the user must enter a value for <i>Attributes</i> of this type <i>frozen</i> (Boolean): if true, the user may not change the value of <i>Attributes</i> of this type
Operations	---

1.4. NumberAttributeType

Description	Concrete <i>AttributeType</i> for <i>NumericalAttributes</i> .
Attributes	<i>predefinedValues</i> (Number[]): an array specifying predefined values for <i>Attributes</i> of this type <i>minValue</i> (Number): the minimum value <i>Attributes</i> of this type may have <i>maxValue</i> (Number): the maximum value <i>Attributes</i> of this type may have
Operations	---

1.5. TextAttributeType

Description	Concrete <i>AttributeType</i> for <i>TextAttributes</i> .
Attributes	<i>predefinedValues</i> (String): an array specifying predefined values for <i>Attributes</i> of this type <i>minSize</i> (Number): the minimum number of characters <i>Attributes</i> of this type may have <i>maxSize</i> (Number): the maximum number of characters <i>Attributes</i> of this type may have
Operations	---

1.6. Attribute

Description	Abstract base class for attributes that a <i>Resource</i> is composed of.
Attributes	---
Operations	---

1.7. NumberAttribute

Description	Concrete <i>Attribute</i> for a numerical property characterisation of a <i>Resource</i> .
Attributes	<i>value</i> (Number): numerical property characterisation of a <i>Resource</i>
Operations	---

1.8. TextAttribute

Description	Concrete <i>Attribute</i> for a textual property characterisation of a <i>Resource</i> .
Attributes	<i>value</i> (String): textual property characterisation of a <i>Resource</i>
Operations	---

1.9. Link

Description	A <i>Link</i> references the two <i>Resources</i> that are linked together by it. Its consistency is checked against the <i>LinkRule</i> it is referenced with.
Attributes	---
Operations	---

1.10. LinkRule

Description	A <i>LinkRule</i> defines characteristics of a consistent <i>Link</i> . It aggregates two <i>LinkRuleEnds</i> .
Attributes	<i>name</i> (String): name of the <i>LinkRule</i> .
Operations	---

1.11. LinkRuleEnd

Description	A <i>LinkRuleEnd</i> belongs to a <i>LinkRule</i> . One <i>LinkRuleEnd</i> of a <i>LinkRule</i> specifies that <i>Resources</i> of the referenced <i>ResourceType</i> may be linked (to <i>Resources</i> of the <i>ResourceType</i> referenced by the other <i>LinkRuleEnd</i>) according to the referenced <i>CardinalitySpec</i> .
Attributes	---
Operations	---

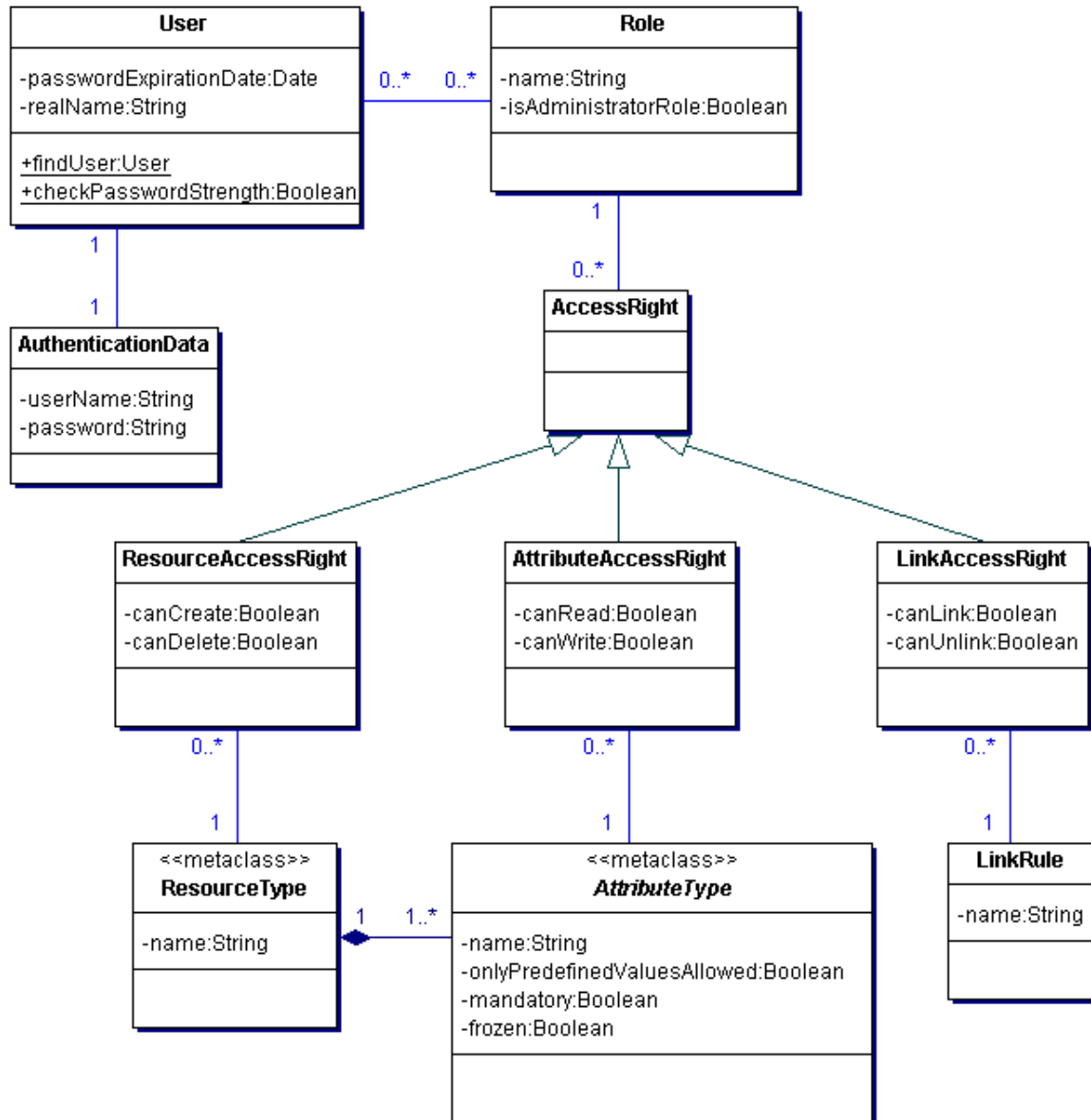
1.12. CardinalitySpec

Description	Specifies the minimum and maximum cardinality for a link rule end of a link rule. Example: A <i>LinkRule</i> has two ends <i>LinkRuleEnd</i> l1 and <i>LinkRuleEnd</i> l2. The <i>LinkRuleEnd</i> l1 references the <i>ResourceType</i> t1 and <i>CardinalitySpec</i> c1 while l2 references t2 and c2. If c1 is min=1 and max=4, this means that one specific <i>Resource</i> of type t2 must have at least 1 and may have up to 4 <i>Links</i> to <i>Resources</i> of type t1. The <i>CardinalitySpec</i> c1 may reference a <i>NumberAttributeType</i> of the <i>ResourceType</i> t2.
Attributes	<i>minCardinality</i> (Number): value for the minimum cardinality; will be ignored when there is a „min“-reference to a <i>NumberAttributeType</i> , in this case the <i>NumberAttribute</i> 's value will be used instead <i>maxCardinality</i> (Number): value for the maximum cardinality; will be ignored when there is a „max“-reference to a <i>NumberAttributeType</i> , in this case the <i>NumberAttribute</i> 's value will be used instead
Operations	---

2. User Management Classes

The following diagram depicts the classes for user management of the MRMS.

Abbildung 2. User Management Classes



2.1. AuthenticationData

Description	Value class, encapsulating the authentication data of a user.
Attributes	<i>userName</i> (String): the user's name <i>password</i> (String): the user's password
Operations	---

2.2. User

Description	Class for user accounts of the MRMS. Its instances may play <i>Roles</i> in the system.
Attributes	<i>passwordExpirationDate</i> (Date): date after which the user has to enter a new

password

realName (String): real name of the user

Operations

- static `checkPasswordStrength(password: String): Boolean`

Effect	Checks, if the given password String is strong enough (minimum length, mixed letters and numbers, ...) to be accepted by the system.
Parameters	<i>password</i> : the password to be checked
Return	The boolean value <i>true</i> , iff the password is strong enough.
Exceptions	---
Actor	Control class of the use case „User changes password“.

- static `findUser(authData: AuthenticationData): User`

Effect	Searches the system for a <i>User</i> matching the given <i>AuthenticationData</i> .
Parameters	<i>authData</i> : the <i>AuthenticationData</i> to search for
Return	If a matching <i>User</i> object could be found it is returned, otherwise the operation returns the <i>null</i> pointer.
Exceptions	---
Actor	Control class of the use case „User logs in“.

2.3. Role

Description	A <i>Role</i> defines which <i>AccessRights</i> its players (<i>Users</i>) have.
Attributes	<i>name</i> (String): name of the <i>Role</i> <i>isAdministratorRole</i> (Boolean): defines if <i>Users</i> of the <i>Role</i> have administration rights
Operations	---

2.4. AccessRight

Description	Abstract base class for access rights. If a <i>Role</i> references an <i>AccessRight</i> it has this <i>AccessRight</i> . <i>Users</i> have the <i>AccessRights</i> which the <i>Roles</i> they play have.
Attributes	---

Operations ---

2.5. ResourceAccessRight

Description	Concrete <i>AccessRight</i> that defines owner's authority of working with <i>Resources</i> that are of a specific <i>ResourceType</i> .
Attributes	<i>canCreate</i> (Boolean): defines if <i>Resources</i> of the referenced <i>ResourceType</i> may be created <i>canDelete</i> (Boolean): defines if <i>Resources</i> of the referenced <i>ResourceType</i> may be deleted
Operations	---

2.6. AttributeAccessRight

Description	Concrete <i>AccessRight</i> that defines owner's authority of working with <i>Attributes</i> of a specific <i>AttributeType</i> that belongs to a specific <i>ResourceType</i> .
Attributes	<i>canRead</i> (Boolean): defines if <i>Attributes</i> of the referenced <i>AttributeType</i> may be read <i>canWrite</i> (Boolean): defines if <i>Attributes</i> of the referenced <i>AttributeType</i> may be written
Operations	---

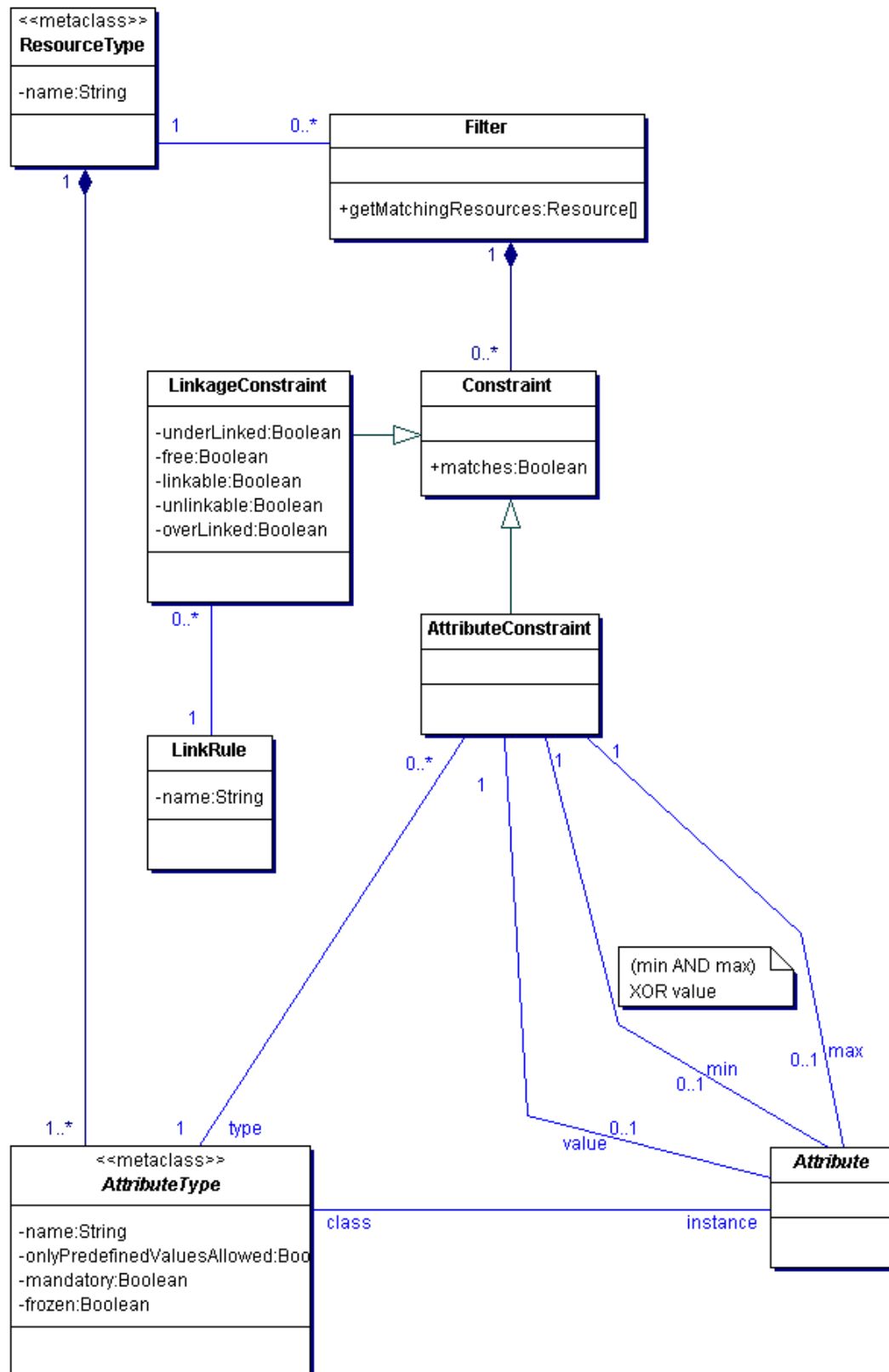
2.7. LinkAccessRight

Description	Concrete <i>AccessRight</i> that defines owner's authority of creating and deleting <i>Links</i> according to a specific <i>LinkRule</i> .
Attributes	<i>canLink</i> (Boolean): defines if <i>Links</i> according to the referenced <i>LinkRule</i> may be created <i>canUnlink</i> (Boolean): defines if <i>Links</i> according to the referenced <i>LinkRule</i> may be deleted
Operations	---

3. Filter Classes

The following diagram depicts the classes needed to create a filtered collection of *Resources*.

Abbildung 3. Filter Classes



3.1. Filter

Description

A *Filter* is used to get a subset of all *Resources* of the referenced *ResourceType*. The

Filter is defined by the *Constraints* it is composed of.

Attributes

Operations

- `getMatchingResources(): Resource[]`

Effect Searches the system for *Resources* matching the referenced *Constraints*.

Parameters ---

Return An array of the matching *Resources*.

Exceptions ---

Actor Control class of the use case „Create filtered collection of resource entries“.

3.2. Constraint

Description Abstract base class for constraints. *Constraints* are used by a *Filter* to describe a specific state that *Resource* must fulfill to pass.

Attributes

Operations

- `matches(resource: Resource): Boolean`

Effect Tests, if the given *Resource* matches this *Constraint*.

Parameters *resource*: the *Resource* to be tested

Return The boolean value *true*, iff the given *Resource* matches this *Constraint*.

Exceptions ---

Actor Class *Filter*.

3.3. AttributeConstraint

Description An *AttributeConstraint* is a concrete *Constraint* that checks whether an *Attribute* of the referenced *AttributeType* is either equal to the referenced *Attribute* or lays between the two referenced min- and max-*Attributes*.

Attributes

Operations

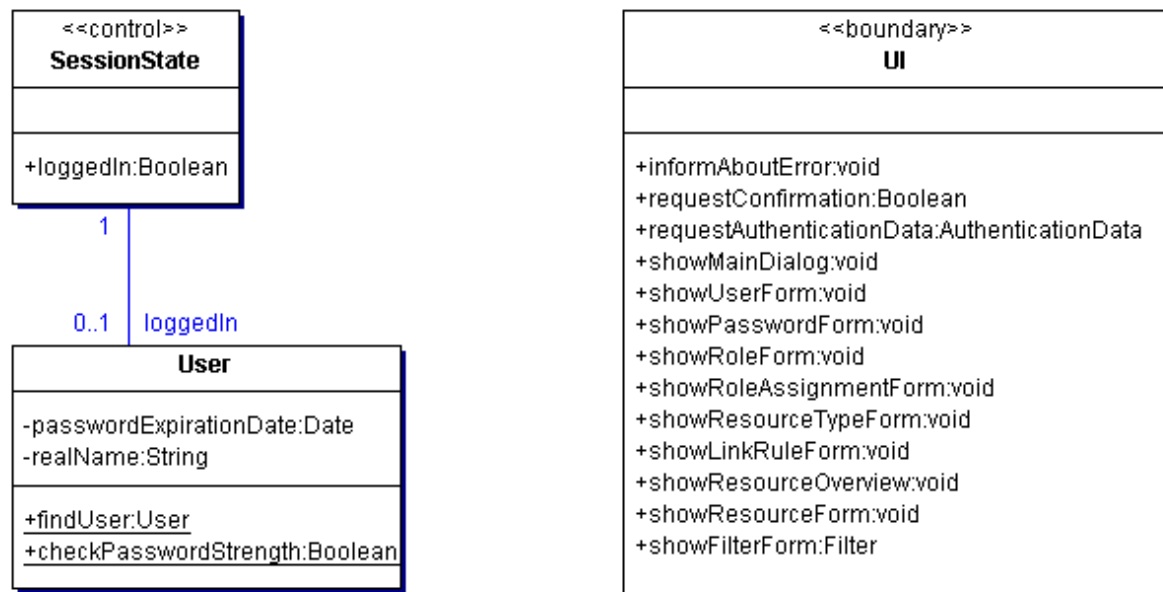
3.4. LinkageConstraint

Description	A <i>LinkageConstraint</i> is a concrete <i>Constraint</i> that checks whether a <i>Resource</i> matches the linkage state that is described by the following attributes. A <i>LinkageConstraint</i> references a <i>LinkRule</i> it refers to: the <i>Link Rule</i> defines a minimum and a maximum cardinality (min and max); the <i>Resource</i> has a current cardinality (cur).
Attributes	<i>underLinked</i> (Boolean): $cur < min$ <i>free</i> (Boolean): $cur = 0$ <i>linkable</i> (Boolean): $cur < max$ <i>unlinkable</i> (Boolean): $cur \geq max$ <i>overLinked</i> (Boolean): $cur > max$
Operations	---

4. Control and Boundary Classes

The following diagram depicts the main classes needed to realize an interaction between the user and the MRMS. (Currently we don't have included the control classes for the use cases.)

Abbildung 4. Control and Boundary Classes



4.1. SessionState

Description	A <i>SessionState</i> describes a session of interaction between the MRMS and a user. A <i>User</i> is logged in in a <i>SessionState</i> if it references that <i>User</i> .
Attributes	---

Operations

- `loggedIn(user: User): Boolean`

Effect	Checks whether the given <i>User</i> is logged in in this <i>SessionState</i> .
Parameters	---
Return	The boolean value <i>true</i> , iff the given <i>User</i> is logged in in this <i>SessionState</i> .
Exceptions	---
Actor	All control classes.

4.2. UI

Description The UI is responsible for providing all facilities the MRMS needs to interact with a user.

Attributes ---

Operations

- `informAboutError(text: String): void`

Effect	The user is informed about an error that occurred in the system.
Parameters	<i>text</i> : the description of the error that occurred
Return	---
Exceptions	---
Actor	Any control class.

- `requestConfirmation(text: String): Boolean`

Effect	The user is requested to confirm or cancel an action.
Parameters	<i>text</i> : the question to be answered by the user
Return	The boolean value <i>true</i> , iff the user confirmed
Exceptions	---
Actor	Any control class.

- `requestAuthenticationData(): AuthenticationData`

Effect	Requests authentication data (username and password) from the user.
--------	---

Parameters	---
Return	An <i>AuthenticationData</i> object holding the values for username and password provided by the user.
Exceptions	<i>AuthenticationAbortedException</i> if the user cancelled the operation
Actor	Control class of the use case „User logs in“.

- `showMainDialog(): void`

Effect	The user is shown the main dialog for interaction with the system.
Parameters	---
Return	---
Exceptions	---
Actor	Main control class.

- `showUserForm(io_user: User): void`

Effect	The user is shown a form for editing a <i>User's</i> attributes.
Parameters	<i>io_user</i> : the <i>User</i> to be edited
Return	---
Exceptions	---
Actor	Control class of the use case „Create user account“.

- `showPasswordForm(io_password: String): void`

Effect	The user is shown a form for editing a <i>User's</i> password.
Parameters	<i>io_password</i> : the password String to be edited
Return	---
Exceptions	---
Actor	Control class of the use case „User changes password“.

- `showRoleForm(io_role: Role): void`

Effect	The user is shown a form for editing a <i>Role's</i> name.
Parameters	<i>io_role</i> : the <i>Role</i> to be edited
Return	---

Exceptions and	---
Actor	Control class of the use case „Create role“.

- showRoleAssignmentForm(io_user: User): void

Effect	The user is shown a form for editing a <i>User</i> 's role assignment.
Parameters	<i>io_user</i> : the <i>User</i> whose role assignment should be edited
Return	---
Exceptions	---
Actor	Control class of the use case „Edit role assignment of user account“.

- showRoleAccessRightsForm(io_role: Role): void

Effect	The user is shown a form for editing a <i>Role</i> 's access rights.
Parameters	<i>io_role</i> : the <i>Role</i> to be edited
Return	---
Exceptions	---
Actor	Control class of the use case „Edit rights for role“.

- showResourceTypeForm(io_resourceType: ResourceType): void

Effect	The user is shown a form for editing a <i>ResourceType</i> 's name and the <i>AttributeTypes</i> it is composed of.
Parameters	<i>io_resourceType</i> : the <i>ResourceType</i> to be edited
Return	---
Exceptions	---
Actor	Control class of the use case „Define resource type“.

- showLinkRuleForm(io_linkRule: LinkRule): void

Effect	The user is shown a form for editing a <i>LinkRule</i> 's attributes.
Parameters	<i>io_linkRule</i> : the <i>LinkRule</i> to be edited
Return	---
Exceptions	---

Actor Control class of the use case „Define link rule“.

- showResourceOverviewForm(resources: Resource[]): void

Effect The user is shown a list of the given *Resources*.

Parameters *resources*: the *Resources* to be shown

Return ---

Exceptions ---

Actor Main control class and control class of the use case „Create filtered collection of resource entries“.

- showResourceForm(io_resource: Resource): void

Effect The user is shown a form for editing a *Resource's Attributes*.

Parameters *io_resource*: the *Resource* to be edited

Return ---

Exceptions ---

Actor Control classes of the use cases „Create resource“ and „Edit resource“.

- showFilterForm(): Filter

Effect The user is shown a form for specifying a *Filter*.

Parameters ---

Return A *Filter* configured according to the users input.

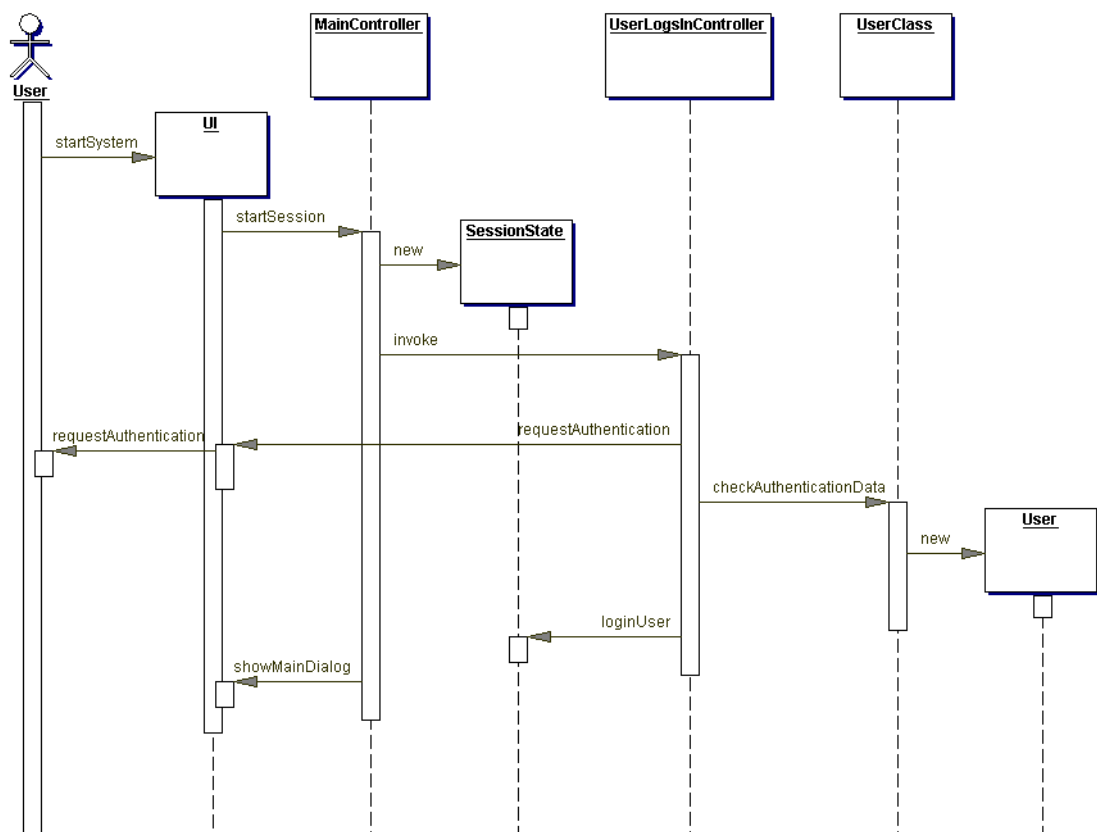
Exceptions ---

Actor Control class of the use case „Create filtered collection of resource entries“.

4.3. Sequence: User logs in

This diagram shows the interactions of a successful log in.

Abbildung 5. User logs in sequence



5. Extensive Overview

Abbildung 6. All Classes

