Lab for Software Engineering

Movie Rating App

Contents

1 Analysis

1.1 A1

1.1.1 Requirements & Domain-Knowledge

Requirements

R1 Users can register with their email address, age, and username. If their age is less than eighteen years old, the registration fails.

R2 After registration, user can login using his data and can also logout afterwards.

R3 Logged-in users can add movies to their list and rate them. If an entered rating differs from the range 1-10, rating process fails, and also an optional comment can be added. Movies without a rating are rated zero per default.

R4 If a movie is not yet in the database, logged-in users can add it.

R5 Registered Users can access a list of all movies in the database sorted by rating in descending order. For each movie, the average rating is calculated and shown together with the comments.

R6 Registered users can add other registered users into a movie discussion group.

R7 A group member can leave a group.

R8 Within a group, members can see movie lists of other members and chat in the group chat and messages are sorted in the chat by the order of their creation.

R9 A group administrator can ban users.

R10 When a group administrator leaves the group, the group is deleted.

R11 If a group only has one member, the group will be automatically be deleted after a certain amount of time.

Facts

F1 Every username is unique.

F2 Each movie has a title, director, at least one character, and original publishing date.

F3 Each movie can be contained in the database once.

F4 A group has a unique name and a list of group members.

F5 A chat message contains the user name of its creator, a time stamp and the content.

F6 The creator of the group is the administrator.

F7 A logged-in user cannot give more than one rating per movie.

Assumptions

A1 Registered Users only add new movies that really exist and movie data that is valid.

A2 While registering users give truthful information about their age and their real email address.

A3 Registered users give fair ratings for movies they have already watched and their ratings are based on their own opinions.

A4 Administrator's decisions are always fair.

A5 Group members only discuss movie-related topics.

1.1.2 Contextdiagram

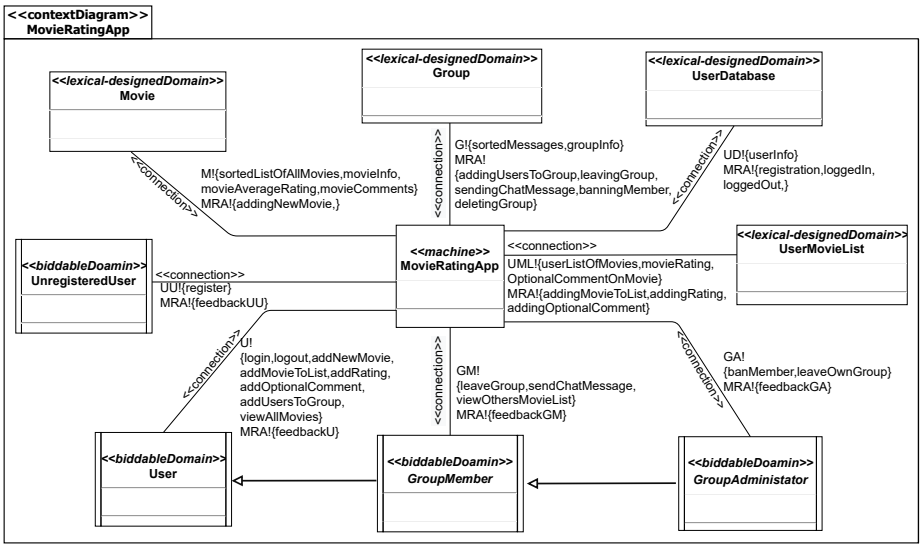

Figure 1.1: Contextdiagram

1.1.3 Validation

- The glossary contains the notions used in R and D.

  The notions mentioned in R and D are contained in the glossary.

- Domains and phenomena of the context diagram must be consistent with R and D.

Notion in CD	Notions in R/D	Type
UnregisteredUser	Users can register	domain
register	After registration, user can login	phenomenon

Continued on next page

– continued from previous page

Notion in CD	Notions in R/D	Type
feedbackUU	introduced to provide feedback to unregistered users	phenomenon
User	Logged-in users can add movies	domain
login	user can login using his data	phenomenon
logout	and can also logout afterwards	phenomenon
addNewMovie	logged-in users can add it	phenomenon
addMovieToList	add movies to their list	phenomenon
addRating	and rate them	phenomenon
addOptionalComment	also an optional comment	phenomenon
addUsersToGroup	add other registered users into a movie discussion group	phenomenon
viewAllMovies	access a list of all movies	phenomenon
feedbackU	introduced to provide feedback to registered users	phenomenon
GroupMember	Within a group, members can see	domain
leaveGroup	A group member can leave a group	phenomenon
sendChatMessage	and chat in the group	phenomenon
viewOthersMovieList	see movie lists of other members	phenomenon
feedbackGM	introduced to provide feedback to registered group members	phenomenon
GroupAdminstrato	The creator of the group is the administrator	domain
leaveOwnGroup	a group administrator leaves	phenomenon
banMember	administrator can ban	phenomenon
feedbackGA	introduced to provide feedback to registered group admins	phenomenon
Movie	If a movie is not yet in the database	domain
sortedListOfAllMovies	list of all movies in the database sorted by rating in descending order	phenomenon
movieInfo	Each movie has a title, director, at least one character, and original publishing date	phenomenon
movieAverageRating	the average rating is calculated	phenomenon
movieComments	shown together with the comments	phenomenon
addingNewMovie	counterpart to addNewMovie	phenomena
Group	into a movie discussion group	domain
groupInfo	A group has a unique name and a list of group members	phenomena
sortedMessages	messages are sorted in the chat	phenomena
addingUsersToGroup	counterpart to addUsersToGroup	phenomena
leavingGroup	counterpart to leaveGroup	phenomena
sendingChatMessage	counterpart to sendChatMessage	phenomena
banningMember	counterpart to banMember	phenomena
deletingGroup	counterpart to leaveOwnGroup	phenomena
UserDatabase	Users can register	domain

Continued on next page

– continued from previous page

Notion in CD	Notions in R/D	Type
Registration	counterpart to register	phenomena
userInfo	Users can register with their email address, age, and username.	phenomena
loggedIn	counterpart to login	phenomena
loggedOut	counterpart to logout	phenomena
UserMovieList	add movies to their list	domain
userListOfMovies	can see movie lists of other members	phenomena
movieRating	and rate them	phenomena
OptionalCommentOnMovie	and also an optional comment can be added	phenomena
addingMovieToList	counterpart to addMovieToList	phenomena
addingRating	counterpart to addRating	phenomena
addingOptionalComment	counterpart to addOptionalComment	phenomena

- There must be exactly one context diagram.

  Only one context diagram is provided.

- A context diagram has at least one machine domain.

  MovieRatingApp is one machine domain.

Domain	Domain Type(s)	Connected Domain(s)	Connected Domain(s) Type(s)
MovieRatingApp	machine domain	Movie	lexical-designed domain
		Group	lexical-designed domain
		UserDatabase	lexical-designed domain
		UnregisteredUser	biddable domain
		UserMovieList	lexical-designed domain
		User	biddable domain
		GroupMember	biddable domain
		GroupAdministrator	biddable domain
UserMovieList	lexical-designed domain	MovieRatingApp	machine domain
Movie	lexical-designed domain	MovieRatingApp	machine domain
Group	lexical-designed domain	MovieRatingApp	machine domain
UserDatabase	lexical-designed domain	MovieRatingApp	machine domain
UnregisteredUser	biddable domain	MovieRatingApp	machine domain
User	biddable domain	MovieRatingApp	machine domain
GroupMember	biddable domain	MovieRatingApp	machine domain
GroupAdministrator	biddable domain	MovieRatingApp	machine domain

- The machine domain must control at least one interface.

  MovieRatingApp controls several interfaces (feedbackU, feedbackRU, . . .).

- Biddable domains cannot be directly connected to lexical domains.

  No biddable domain is connected to a lexical domain.

Domain	Domain Type(s)	Connected Domain(s)	Connected Domain(s) Type(s)
MovieRatingApp	machine domain	Movie	lexical-designed domain
		Group	lexical-designed domain
		UserDatabase	lexical-designed domain
		UnregisteredUser	biddable domain
		UserMovieList	lexical-designed domain
		User	biddable domain
		GroupMember	biddable domain
		GroupAdministrator	biddable domain
UserMovieList	lexical-designed domain	MovieRatingApp	machine domain
Movie	lexical-designed domain	MovieRatingApp	machine domain
Group	lexical-designed domain	MovieRatingApp	machine domain
UserDatabase	lexical-designed domain	MovieRatingApp	machine domain
UnregisteredUser	biddable domain	MovieRatingApp	machine domain
User	biddable domain	MovieRatingApp	machine domain
GroupMember	biddable domain	MovieRatingApp	machine domain
GroupAdministrator	biddable domain	MovieRatingApp	machine domain

- Causal, designed, lexical, display, machine domain type are not allowed together with biddable domain.

  User, RegisteredUser, GroupMember, GroupAdministrator are biddable domains only.

Domain	Domain Type(s)	Connected Domain(s)	Connected Domain(s) Type(s)
MovieRatingApp	machine domain	Movie	lexical-designed domain
		Group	lexical-designed domain
		UserDatabase	lexical-designed domain
		UnregisteredUser	biddable domain
		UserMovieList	lexical-designed domain
		User	biddable domain
		GroupMember	biddable domain
		GroupAdministrator	biddable domain
UserMovieList	lexical-designed domain	MovieRatingApp	machine domain
Movie	lexical-designed domain	MovieRatingApp	machine domain
Group	lexical-designed domain	MovieRatingApp	machine domain
UserDatabase	lexical-designed domain	MovieRatingApp	machine domain
UnregisteredUser	biddable domain	MovieRatingApp	machine domain
User	biddable domain	MovieRatingApp	machine domain
GroupMember	biddable domain	MovieRatingApp	machine domain
GroupAdministrator	biddable domain	MovieRatingApp	machine domain

- Phenomena controlled by a biddable domain must have counterpart phenomena located between machine and causal/lexical/designed domains.

	biddable domain phenomena	counterpart
User	register	registration
RegisteredUser		
	login	loggedIn
	logout	loggedOut

Name	Type	Description
	addNewMovie	addingNewMovie
	addMovieToList	addingMovieToList
	addRating	addingRating
	addOptionalComment	addingOptionalComment
	addUsersToGroup	addingUsersToGroup
	viewAllMovies	sortedListOfAllMovies
GroupMember		
	leaveGroup	leavingGroup
	sendChatMessage	sendingChatMessage
	viewOthersMovieList	userMovieList
GroupAdminstrator		
	banMember	banningMember
	leaveOwnGroup	deletingGroup

- Connection domains must have at least one observed and one controlled interface.
- For each phenomenon controlled by a connection domain, there must be at least one phenomenon controlled by one of the connected domains, i.e. observed by the connection domain.
- For each phenomenon observed by a connection domain, there must be at least one phenomenon controlled the connection domain, i.e. for each input there is an output.

  Context diagram contains no connection domain.

1.2 A2

1.2.1 Problem Diagrams & Mapping

R1 Users can register with their email address, age, and username. If their age is less than eighteen years old, the registration fails.

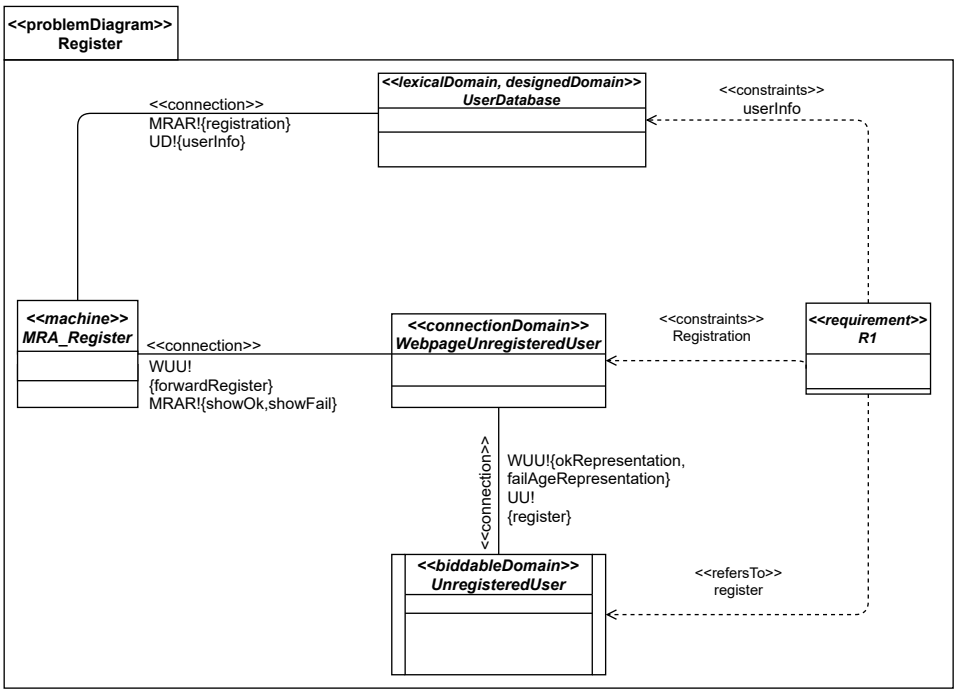

Figure 1.2: Problemdiagram for R1

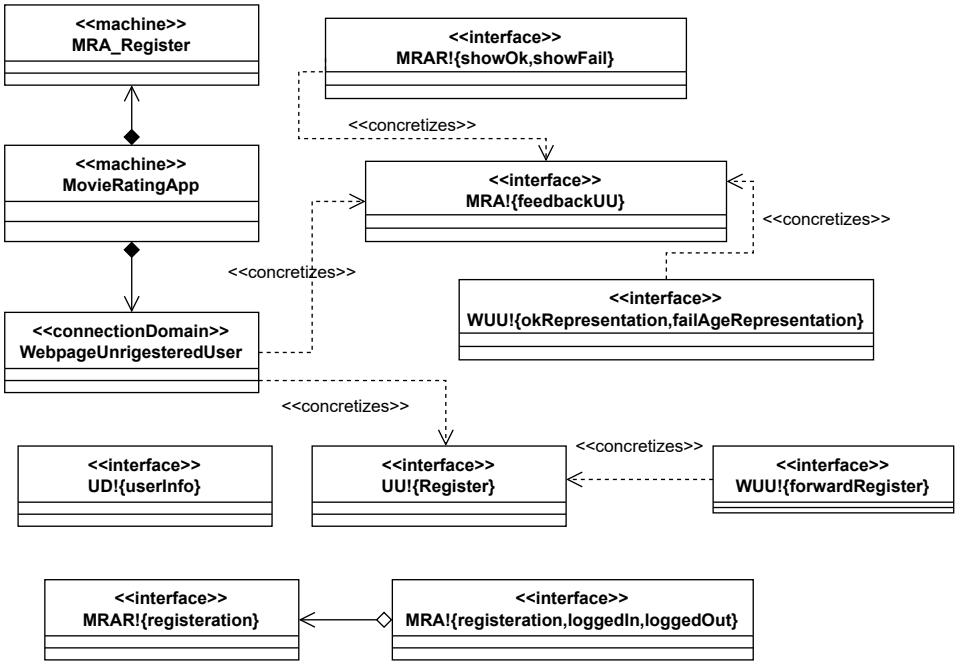

Figure 1.3: Mapping for PD1

R3 Logged-in users can add movies to their list and rate them. If an entered rating differs from the range 1-10, rating process fails, and also an optional comment can be added. Movies without a rating are rated zero per default.

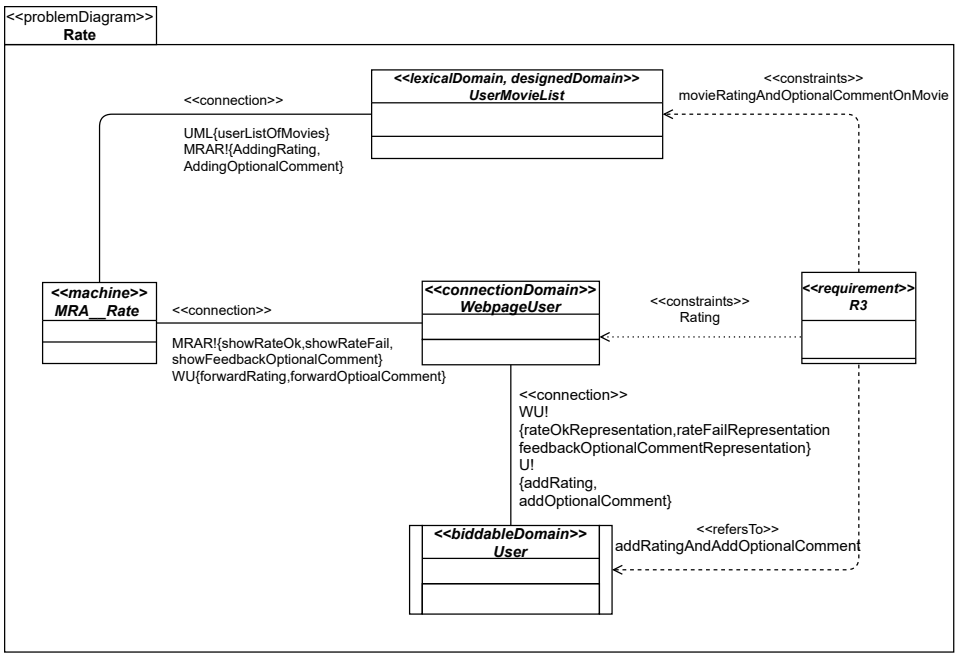

Figure 1.4: Problemdiagram for R3

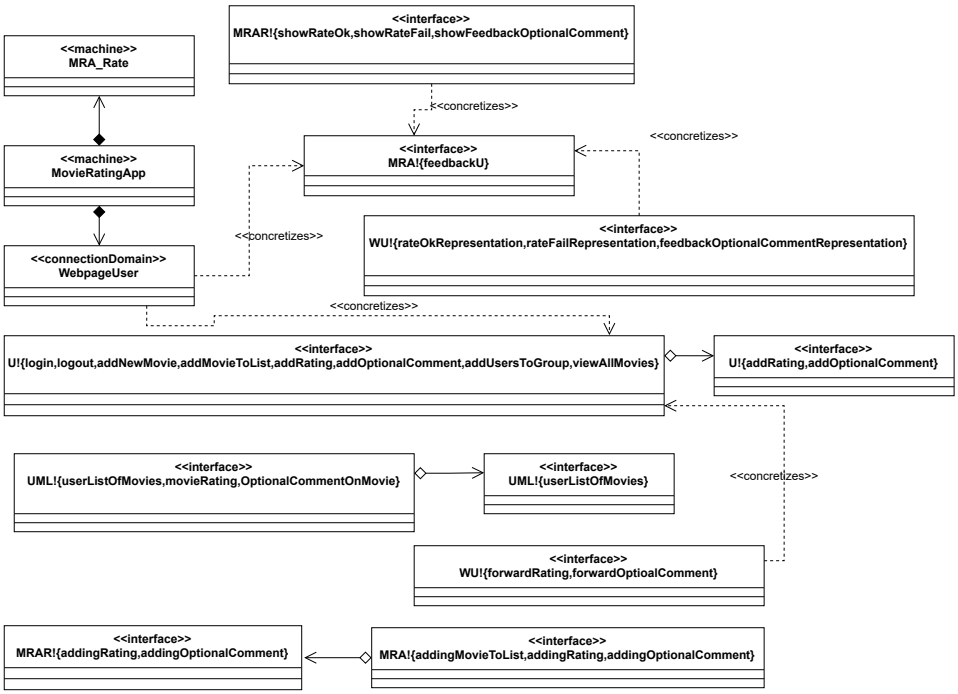

Figure 1.5: Mapping for PD3

R4 If a movie is not yet in the database, logged-in users can add it.

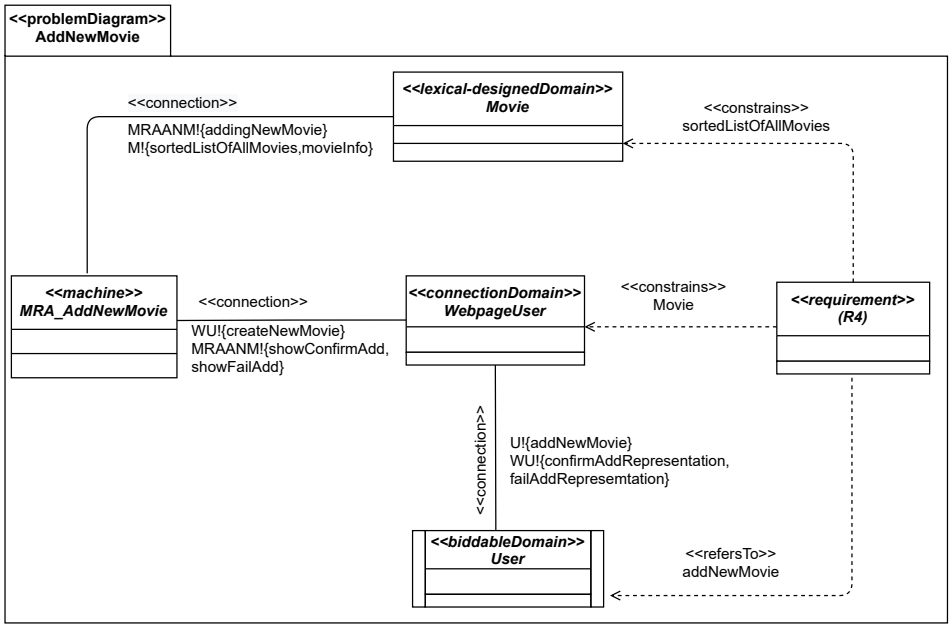

Figure 1.6: Problemdiagram for R4

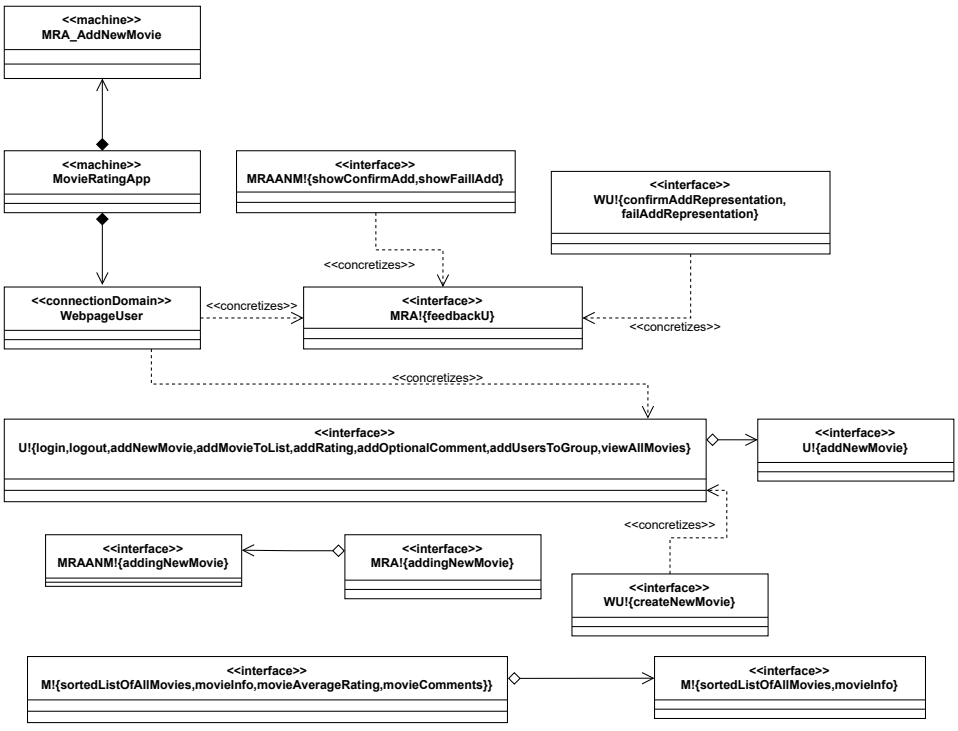

Figure 1.7: Mapping for PD4

R5 Registered Users can access a list of all movies in the database sorted by rating in descending order. For each movie, the average rating is calculated and shown together with the comments.

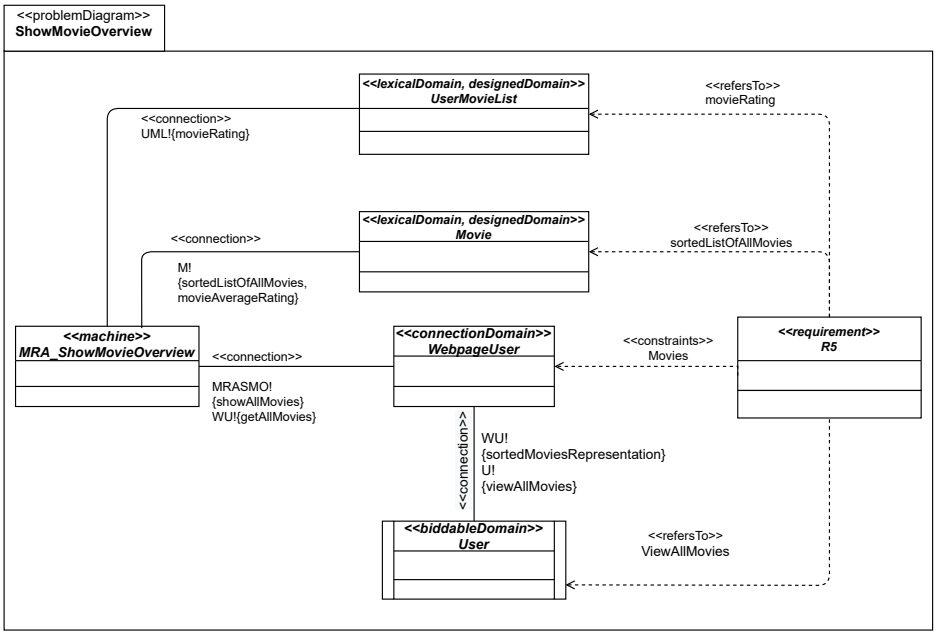

Figure 1.8: Problemdiagram for R5

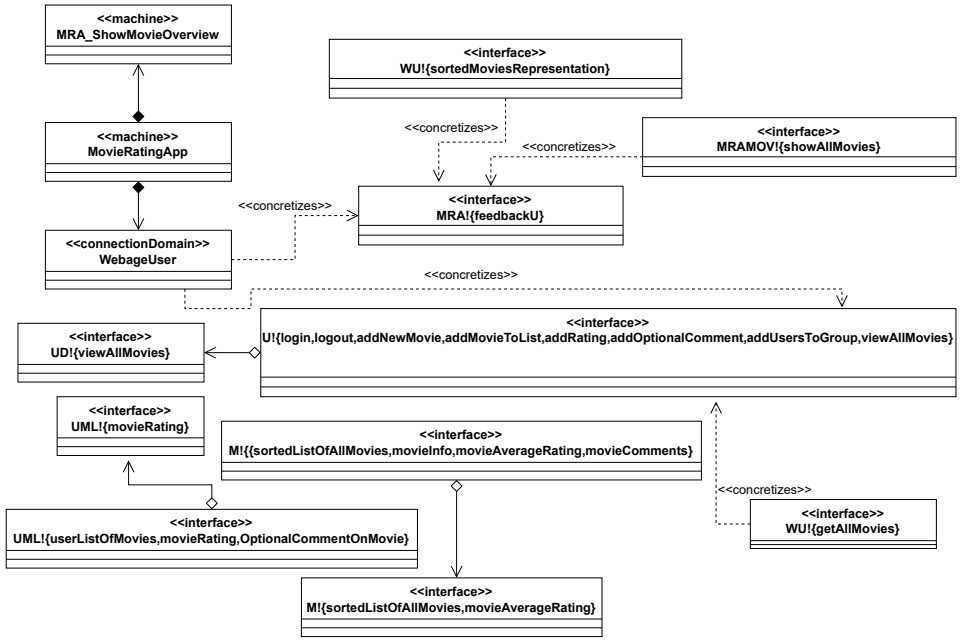

Figure 1.9: Mapping for PD5

1.2.2 Validation

- All relevant requirements R are covered in some subproblem.

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdRegister	MRA_Register	machine		registration, showOk,showFail
		UserDatabase	lexical, designed	X	userInfo
		WebpageUnregisteredUser	connection	X	forwardRegister, okRepresentation, failAgeRepresentation
		UnregisteredUser	biddable		register
R03	pdRate	MRA_Rate	machine		addingRating, addingOptionalComment, showRateOk, showRateFail, showFeedbackOptionalComment
		UserMovieList	lexical, designed	X	userListOfMovies
		WebpageUser	connection	X	forwardRating, forwardOptionalComment, rateOkRepresentation, rateFailRepresentation, feedbackOptionalCommentRepresentation
		User	biddable		addRating, addOptionalComment
R04	pdAddNewMovie	MRA_AddNewMovie	machine		addingNewMovie, showConfirmAdd, showFailAdd
		Movie	lexical, designed	X	sortedListOfAllMovies, movieInfo
		WebpageUser	connection	X	createNewMovie, confirmAddRepresentation, failAddRepresentation
		User	biddable		addNewMovie
R05	pdShowMovieOverview	MRA_ShowMovieOverview	machine		showAllmovies
		Movie	lexical, designed		sortedListOfAllMovies, movieAverageRating
		UserMovieList	lexical, designed		movieRating
		WebpageUser	connection	X	getAllMovies, sortedMoviesRepresentation,
		User	biddable		viewAllMovies

- A problem diagram has exactly one machine domain.

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdRegister	MRA_Register	machine		registration, showOk,showFail
		UserDatabase	lexical, designed	X	userInfo
		WebpageUnregisteredUser	connection	X	forwardRegister, okRepresentation, failAgeRepresentation
		UnregisteredUser	biddable		register
R03	pdRate	MRA_Rate	machine		addingRating, addingOptionalComment, showRateOk, showRateFail, showFeedbackOptionalComment
		UserMovieList	lexical, designed	X	userListOfMovies
		WebpageUser	connection	X	forwardRating, forwardOptionalComment, rateOkRepresentation, rateFailRepresentation, feedbackOptionalCommentRepresentation
		User	biddable		addRating, addOptionalComment
R04	pdAddNewMovie	MRA_AddNewMovie	machine		addingNewMovie, showConfirmAdd, showFailAdd
		Movie	lexical, designed	X	sortedListOfAllMovies, movieInfo
		WebpageUser	connection	X	createNewMovie, confirmAddRepresentation, failAddRepresentation
		User	biddable		addNewMovie
R05	pdShowMovieOverview	MRA_ShowMovieOverview	machine		showAllmovies
		Movie	lexical, designed		sortedListOfAllMovies, movieAverageRating
		UserMovieList	lexical, designed		movieRating
		WebpageUser	connection	X	getAllMovies, sortedMoviesRepresentation,
		User	biddable		viewAllMovies

- A problem diagram contains at least one requirement.

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdRegister	MRA_Register	machine		registration, showOk,showFail
		UserDatabase	lexical, designed	X	userInfo
		WebpageUnregisteredUser	connection	X	forwardRegister, okRepresentation, failAgeRepresentation
		UnregisteredUser	biddable		register
R03	pdRate	MRA_Rate	machine		addingRating, addingOptionalComment, showRateOk, showRateFail, showFeedbackOptionalComment
		UserMovieList	lexical, designed	X	userListOfMovies
		WebpageUser	connection	X	forwardRating, forwardOptionalComment, rateOkRepresentation, rateFailRepresentation, feedbackOptionalCommentRepresentation
		User	biddable		addRating, addOptionalComment
R04	pdAddNewMovie	MRA_AddNewMovie	machine		addingNewMovie, showConfirmAdd, showFailAdd
		Movie	lexical, designed	X	sortedListOfAllMovies, movieInfo
		WebpageUser	connection	X	createNewMovie, confirmAddRepresentation, failAddRepresentation
		User	biddable		addNewMovie
R05	pdShowMovieOverview	MRA_ShowMovieOverview	machine		showAllmovies
		Movie	lexical, designed		sortedListOfAllMovies, movieAverageRating
		UserMovieList	lexical, designed		movieRating
		WebpageUser	connection	X	getAllMovies, sortedMoviesRepresentation,
		User	biddable		viewAllMovies

- The machine domain must control at least one interface.

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdRegister	MRA_Register	machine		registration, showOk,showFail
		UserDatabase	lexical, designed	X	userInfo
		WebpageUnregisteredUser	connection	X	forwardRegister, okRepresentation, failAgeRepresentation
		UnregisteredUser	biddable		register
R03	pdRate	MRA_Rate	machine		addingRating, addingOptionalComment, showRateOk, showRateFail, showFeedbackOptionalComment
		UserMovieList	lexical, designed	X	userListOfMovies
		WebpageUser	connection	X	forwardRating, forwardOptionalComment, rateOkRepresentation, rateFailRepresentation, feedbackOptionalCommentRepresentation
		User	biddable		addRating, addOptionalComment
R04	pdAddNewMovie	MRA_AddNewMovie	machine		addingNewMovie, showConfirmAdd, showFailAdd
		Movie	lexical, designed	X	sortedListOfAllMovies, movieInfo
		WebpageUser	connection	X	createNewMovie, confirmAddRepresentation, failAddRepresentation
		User	biddable		addNewMovie
R05	pdShowMovieOverview	MRA_ShowMovieOverview	machine		showAllmovies
		Movie	lexical, designed		sortedListOfAllMovies, movieAverageRating
		UserMovieList	lexical, designed		movieRating
		WebpageUser	connection	X	getAllMovies, sortedMoviesRepresentation,
		User	biddable		viewAllMovies

- Requirements constrain at least one domain.

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdRegister	MRA_Register	machine		registration, showOk,showFail
		UserDatabase	lexical, designed	X	userInfo
		WebpageUnregisteredUser	connection	X	forwardRegister, okRepresentation, failAgeRepresentation
		UnregisteredUser	biddable		register
R03	pdRate	MRA_Rate	machine		addingRating, addingOptionalComment, showRateOk, showRateFail, showFeedbackOptionalComment
		UserMovieList	lexical, designed	X	userListOfMovies
		WebpageUser	connection	X	forwardRating, forwardOptionalComment, rateOkRepresentation, rateFailRepresentation, feedbackOptionalCommentRepresentation
		User	biddable		addRating, addOptionalComment
R04	pdAddNewMovie	MRA_AddNewMovie	machine		addingNewMovie, showConfirmAdd, showFailAdd
		Movie	lexical, designed	X	sortedListOfAllMovies, movieInfo
		WebpageUser	connection	X	createNewMovie, confirmAddRepresentation, failAddRepresentation
		User	biddable		addNewMovie
R05	pdShowMovieOverview	MRA_ShowMovieOverview	machine		showAllmovies
		Movie	lexical, designed		sortedListOfAllMovies, movieAverageRating
		UserMovieList	lexical, designed		movieRating
		WebpageUser	connection	X	getAllMovies, sortedMoviesRepresentation,
		User	biddable		viewAllMovies

- Requirements do not constrain machine(s).

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdRegister	MRA_Register	machine		registration, showOk,showFail
		UserDatabase	lexical, designed	X	userInfo
		WebpageUnregisteredUser	connection	X	forwardRegister, okRepresentation, failAgeRepresentation
		UnregisteredUser	biddable		register
R03	pdRate	MRA_Rate	machine		addingRating, addingOptionalComment, showRateOk, showRateFail, showFeedbackOptionalComment
		UserMovieList	lexical, designed	X	userListOfMovies
		WebpageUser	connection	X	forwardRating, forwardOptionalComment, rateOkRepresentation, rateFailRepresentation, feedbackOptionalCommentRepresentation
		User	biddable		addRating, addOptionalComment
R04	pdAddNewMovie	MRA_AddNewMovie	machine		addingNewMovie, showConfirmAdd, showFailAdd
		Movie	lexical, designed	X	sortedListOfAllMovies, movieInfo
		WebpageUser	connection	X	createNewMovie, confirmAddRepresentation, failAddRepresentation
		User	biddable		addNewMovie
R05	pdShowMovieOverview	MRA_ShowMovieOverview	machine		showAllmovies
		Movie	lexical, designed		sortedListOfAllMovies, movieAverageRating
		UserMovieList	lexical, designed		movieRating
		WebpageUser	connection	X	getAllMovies, sortedMoviesRepresentation,
		User	biddable		viewAllMovies

- If requirements do constrain biddable domains, a good argument is given and documented.

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R01	pdRegister	MRA_Register	machine		registration, showOk,showFail
		UserDatabase	lexical, designed	X	userInfo
		WebpageUnregisteredUser	connection	X	forwardRegister, okRepresentation, failAgeRepresentation
		UnregisteredUser	biddable		register
R03	pdRate	MRA_Rate	machine		addingRating, addingOptionalComment, showRateOk, showRateFail, showFeedbackOptionalComment
		UserMovieList	lexical, designed	X	userListOfMovies
		WebpageUser	connection	X	forwardRating, forwardOptionalComment, rateOkRepresentation, rateFailRepresentation, feedbackOptionalCommentRepresentation
		User	biddable		addRating, addOptionalComment
R04	pdAddNewMovie	MRA_AddNewMovie	machine		addingNewMovie, showConfirmAdd
		Movie	lexical, designed	X	sortedListOfAllMovies, movieInfo
		WebpageUser	connection	X	createNewMovie, confirmAddRepresentation, failAddRepresentation
		User	biddable		addNewMovie
R05	pdShowMovieOverview	MRA_ShowMovieOverview	machine		showAllmovies
		Movie	lexical, designed		sortedListOfAllMovies, movieAverageRating
		UserMovieList	lexical, designed		movieRating
		WebpageUser	connection	X	getAllMovies, sortedMoviesRepresentation,
		User	biddable		viewAllMovies

- Connection domains must have at least one observed and one controlled interface.

connection domain	phenomenon controlled by connection domain	connected Domain	phenomenon controlled by connected domain
Webpage-UnregisteredUser	forwardRegister	UnregisteredUser	register
	okRepresentation	MRA_Register	showOk
	failAgeRepresentation	MRA_Register	showFail
WebpageUser	createNewMovie	User	addNewMovie
	confirmAddRepresentation	MRA_AddNewMovie	showConfirmAdd
	failAddRepresentation	MRA_AddNewMovie	showFailAdd
	forwardRating	User	addRating
	rateOkRepresentation	MRA_Rate	showRateOk
	rateFailRepresentation	MRA_Rate	showRateFail
	forwardOptioalComment	User	addOptionalComment
	feedbackOptional-CommentRepresentation	MRA_Rate	showFeedback-OptionalComment
	getAllMovies	User	viewAllMovies
	sortedMovies-Representation	MRA_ShowMovieOverview	showAllMovies

- For each phenomenon controlled by a connection domain, there must be at least one phenomenon controlled by one of the connected domains, i.e. observed by the connection domain.

connection domain	phenomenon controlled by connection domain	connected Domain	phenomenon controlled by connected domain
Webpage-UnregisteredUser	forwardRegister	UnregisteredUser	register
	okRepresentation	MRA_Register	showOk
	failAgeRepresentation	MRA_Register	showFail
WebpageUser	createNewMovie	User	addNewMovie
	confirmAddRepresentation	MRA_AddNewMovie	showConfirmAdd
	failAddRepresentation	MRA_AddNewMovie	showFailAdd
	forwardRating	User	addRating
	rateOkRepresentation	MRA_Rate	showRateOk
	rateFailRepresentation	MRA_Rate	showRateFail
	forwardOptioalComment	User	addOptionalComment
	feedbackOptional-CommentRepresentation	MRA_Rate	showFeedback-OptionalComment
	getAllMovies	User	viewAllMovies
	sortedMovies-Representation	MRA_ShowMovieOverview	showAllMovies

- For each phenomenon observed by a connection domain, there must be at least one phenomenon controlled by the connection domain, i.e. for each input there is an output.

connection domain	phenomenon observed by connection domain	phenomenon controlled by connection domain
WebpageUnregisteredUser	register	forwardregister
	showOk	okRepresentation
	showFail	failAgeRepresentation
WebpageUser	addNewMovie	createNewMovie
	showConfirmAdd	confirmAddRepresentation
	showFailAdd	failAddRepresentation
	addRating	forwardRating
	showRateOk	ratefailRepresentation
	showRateFail	rateFailRepresentation
	addOptionalComment	forwardOptionalComment
	showFeedbackOptionalComment	feedbackOptionalCommentRepresentation
	viewAllMovies	getAllMovies
	showAllMovies	sortedMoviesRepresentation

- The problem diagrams must be consistent to the context diagram, e.g. each machine of the problem diagrams is a part of the context diagram machine.

\+ *Provided mapping diagrams*

- All subproblems can be derived from the context diagram by means of decomposition operators.

problem Diagram	operators	related domains or phenomena
pdRegister	leave out domain	Movie,UserMovieList, User,Group, GroupMember, GroupAdministrator
	Introduce connection/display domain	WebpageUnregisteredUser
	split Interface	MRA!{...}
	concretize interface	MRA!{...}, UU!{...}
pdAddNewMovie	leave out domain	UserMovieList,UserDatabase, UnregisteredUser, Group,GroupMember, GroupAdministrator
	Introduce connection/display domain	WebpageUser
	split Interface	MRA!{...} ,M!{...}
	concretize interface	MRA!{...}, U!{...}
pdRate	leave out domain	Movie,Group, UnregisteredUser,Userdatabase, GroupMember,GroupAdministrator
	Introduce connection/display domain	WebpageUser

Continued on next page

– continued from previous page

problem Diagram	operators	related domains or phenomena
	split Interface	MRA!{...},UML!{...}
	concretize interface	MRA!{...},U!{...}
pdShowMovieOverview	leave out domain	UserDatabase,UnregisteredUser, Group,GroupMember, GroupAdministrator
	Introduce connection/display domain	WebpageUser
	split Interface	MRA!{...},M!{...},UML!{...}
	concretize interface	MRA!{...}, U!{...}

+ *Provided mapping diagrams*

1.2.3 Problem Frames

Subproblem for requirement(s)

R1 (Register) in figure 1.2.1 fits to **Update(2)**

R3 (Rate) in figure 1.4- fits to **Update(2)**

R4 (AddNewMovie) in figure 1.6- fits to **Update(2)**

R5 (ShowMovieOverview) in figure 1.8- fits to **Query(2)** variant

1.2.4 Validation for Problem Frames

- All connections in a problem diagram correspond to a connection in the frame diagram (connects same domain types).

Problem Diagram	**Problem Frame**	**Connections in PD**	**Connections in PF**	**Domain Type 1**	**Domain Type 2**
Register	update2	MRAR!{registration} UD!{userInfo}	UM!Y2, DB!Y1	Machine	LexicalDomain
		MRAR!{showOk,showFail} WUU!{forwardRegister}	UM!E4, IOD!E8	Machine	ConnectionDomain
		WUU!{okRepresentation, failAgeRepresentation} UU!{register}	IOD!C7, UO!E6	ConnectionDomain	BiddableDomain
Rate	update2	MRAR!{AddingRating, AddingOptionalComment} UML{userListOfMovies}	UM!Y2, DB!Y1	Machine	LexicalDomain
		MRAR!{showRateOk, showRateFail,showFeedback-OptionalComment} WU!{forwardRegister}	UM!E4, IOD!E8	Machine	ConnectionDomain
		WU!{rateOkRepresentation, rateFailRepresentation, feed-backOptionalCommentRepre-sentation} U!{register}	IOD!C7, UO!E6	ConnectionDomain	BiddableDomain
AddNew-Movie	update2	MRAANM!{addingNew-Movie} M!{sortedListOfAllMovies, movieInfo}	UM!Y2, DB!Y1	Machine	LexicalDomain

Continued on next page

– continued from previous page

Problem Diagram	Problem Frame	Connections in PD	Connections in PF	Domain Type 1	Domain Type 2
		MRAANM!{showConfirm-Add,showFaillAdd} WU!{createNewMovie} WU!{confirmAddRepresent-ation,faillAddRepresentation} U!{addNewMovie}	UM!E4, IOD!E8 IOD!C7, UO!E6	Machine ConnectionDomain	ConnectionDomain BiddableDomain
ShowMovie-Overview	query2 (User-MovieList &Movie merged)	UML!{movieRating} M{sortedListOfAllMovies, movieAverageRating}	DB!Y1	Machine	LexicalDomain
		MRASMO!{showAllMovies, } WU!{getAllMovies}	QM!Y1, IOD!C6	Machine	ConnectionDomain
		WU!{sortedMoviesRepresent-ation,} U!{viewAllMovies}	IOD!C7, EO!E5	ConnectionDomain	BiddableDomain

- The domain types of constrained domains in the problem diagram are the same as in the frame diagram.

Problem Diagram	Problem Frame	Constrained Domains in PD	Constrained Domains in PF	Domain Type
Register	update 2	UserDatabase	Data Base	LexicalDomain
		WebpageUnregisteredUser	Input Output Device	ConnectionDomain
AddNewMovie	update 2	Movie	Data Base	LexicalDomain
		WebpageUser	Input Output Device	ConnectionDomain
Rate	update 2	UserMovieList	Data Base	LexicalDomain
		WebpageUser	Input Output Device	ConnectionDomain
ShowMovieOverview	query 2	WebpageUser	Input Output Device	ConnectionDomain

- Each referred domain in the problem frame corresponds to a domain in the problem diagram.

Problem Diagram	Problem Frame	Referred Domains in PD	Referred Domains in PF	Domain Type
Register	update 2	UnregisteredUser	Update Operator	BiddableDomain
AddNewMovie	update 2	User	Update Operator	BiddableDomain
Rate	update 2	User	Update Operator	BiddableDomain
ShowMovieOverview	query 2	User	Enquiry Operator	BiddableDomain
		Movie	Database	lexicalDomain
		UserMovieList	Database	lexicalDomain

■ *All problem diagrams are consistent to their problem frames.*

1.3 A3

1.3.1 Abstract Software Specifications & Sequence Diagram

(R1) Users can register with their email address, age, and username. If their age is less than eighteen years old, the registration fails.

- Using the domain knowledge

(F1) Every username is unique.

- we can derive the specifications:

WebpageUU(S1a) When the webpage receives the command "register", then the command is forwarded to the machine with the command "forwardRegister". The feedback whether the registration process was successful or not is received via the commands "showOK" and "showFail". Both feedback types are shown to the unregistered user via the commands "okRepresentation" and "failAgeRepresentation".

MRA_Register(S1b) When the machine receives the command "forwardRegister", then it is checked if the username entered by user is available with the command "get_userInfo" and the result is received as the data "userInfo". If the users age is eighteen or more, then the machine sends command "registration", and informs the unregistered user with the command "showOk" to the webpage that the registration is successful. Otherwise, the machine informs the unregistered user with the command "showFail" to the web page about the unsuccessful registration process.

UserDatabase(S1c) After receiving the command "get_userInfo" the results are returned as the data "userInfo". When the command "registration" is received, the users info is added to the UserDatabase.

Correctness condition: $(S1a) \wedge (S1b) \wedge (S1c) \wedge (F1) \implies (R1)$

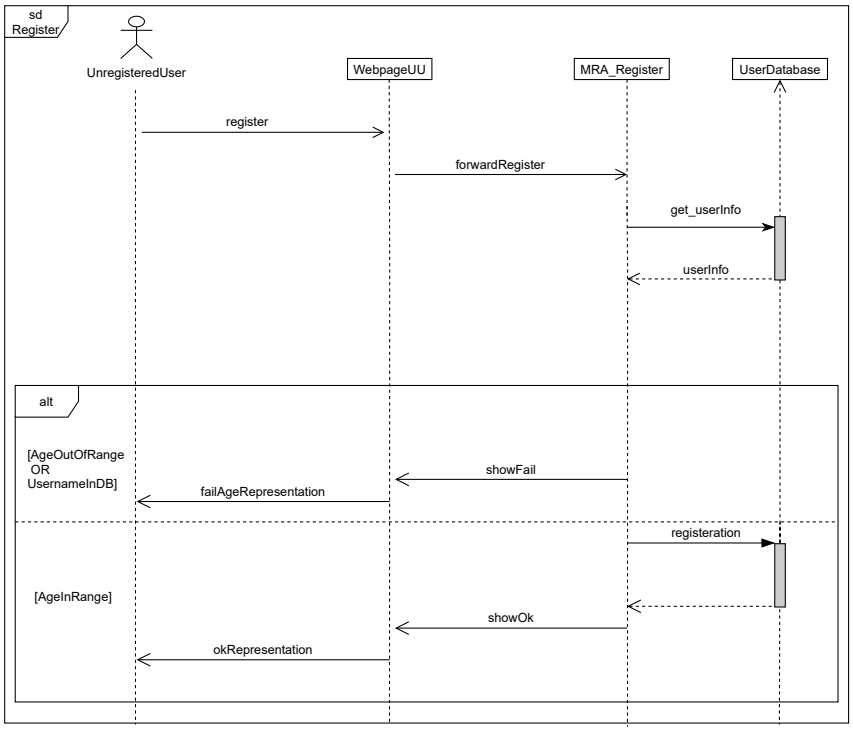

Figure 1.10: SequenceDiagram for R1

R3 Logged-in users can add movies to their list and rate them. If an entered rating differs from the range 1-10, rating process fails, and also an optional comment can be added. Movies without a rating are rated zero per default.

- Using the domain knowledge

(F7) A logged-in user cannot give more than one rating per movie.

(F1) Every username is unique.

(A3) Registered users give fair ratings for movies they have already watched and their ratings are based on their own opinions.

- we can derive the specifications:

WebpageU(S3a) When the web page receives the command "addRating", then the rating is forwarded to the machine with command "forwardRating". The feedback is received via the commands "showRateOk" or "showRateFail" and then shown to the user by "rateOkRepresentation" or"rateFailRepresentation" and similarly when the web page receives addOptionalComment it is forwarded to the machine with forwardOptionalComment and then feedback is received via the commands "ShowFeedbackOptionalCommentRepresentation" and then show to the user by "feedbackOptionalCommentRepresentation" .

MRA_Rate(S3b) When the machine receives the command "forwardRating", then it is checked if the user added a rating before with the command "get_userListOfMovies" and the result is received as the data "userListOfMovies". If the rating is in range, then the machine sends command "addingRating", and informs the unregistered user with the command "showRateOk" to the webpage that the rating is added successful. optionally, if the user wants to add a comment afterwards the comment is received by "forwardOptioalComment" and then sent to database with "addingOptionalComment". Otherwise, the machine informs the user with the command "showRateFail" to the web page about the unsuccessful rating process.

UserMovieList(S3c) When the database receives command "get_userListOfMovies" it returns "userListOfMovies". When the commands "AddingRating" and "AddingOptionalComment" is received, the rating and the optional comment is added in the users movie list .

Correctness condition: $(S3a) \wedge (S3b) \wedge (S3c) \wedge (F7) \wedge (F1) \wedge (A3) \implies (R3)$

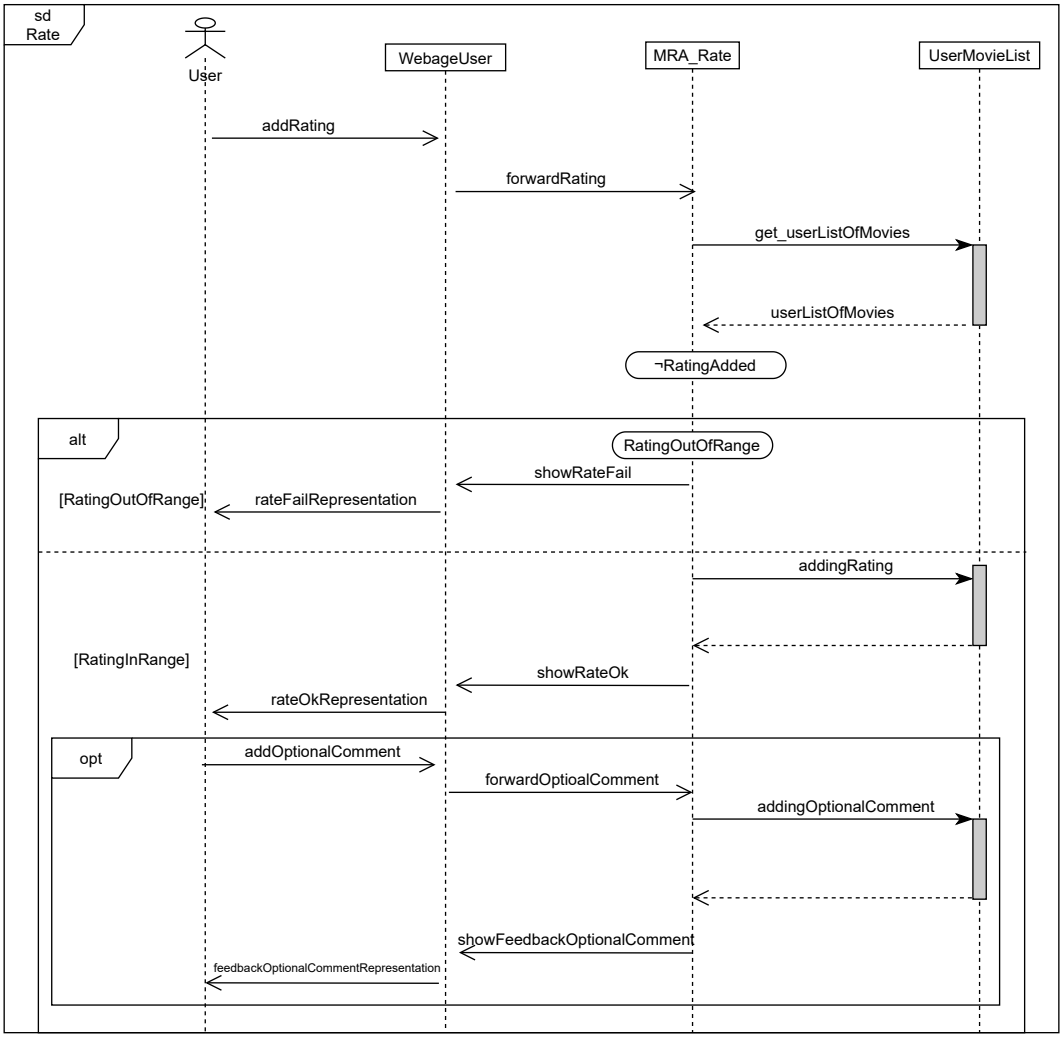

Figure 1.11: SequenceDiagram for R3

R4 If a movie is not yet in the database, logged-in users can add it.

- Using the domain knowledge

F3 Each movie can be contained in the database once.

F2 Each movie has a title, director, at least one character, and original publishing date.

- we can derive the specifications:

WebpageU(S4a) When the webpage receives the command "addNewMovie", then the command is forwarded to the machine with the command "createNewMovie". The feedback is received via the commands "showConfirmAdd" in case it was successfully added to the database and shown to the user by "confirmAddRepresentation", or"showFailAdd" in case it failed to add it to the database and shown to the user by "failAddRepresentation".

MRA_AddNewMovie(S4b) When the machine receives the command "createNewMovie", "get_sortedListOffAllMovies" is sent to check if the movie is in database or not the results are turned as "sortedListOffAllMovies". afterwards an entry is created in the Movie database with the command "addingNewMovie" The confirmation is sent the web page via the command "showConfirmAdd".

Movie(S5c) When the command "get_sortedListOffAllMovies" is recived, "sortedListOffAllMovies" is returned. and when "addingNewMovie" is received, the new movie is added in the database.

Correctness condition: $(S4a) \wedge (S4b) \wedge (S4c) \wedge (F3) \wedge (F2) \implies (R4)$

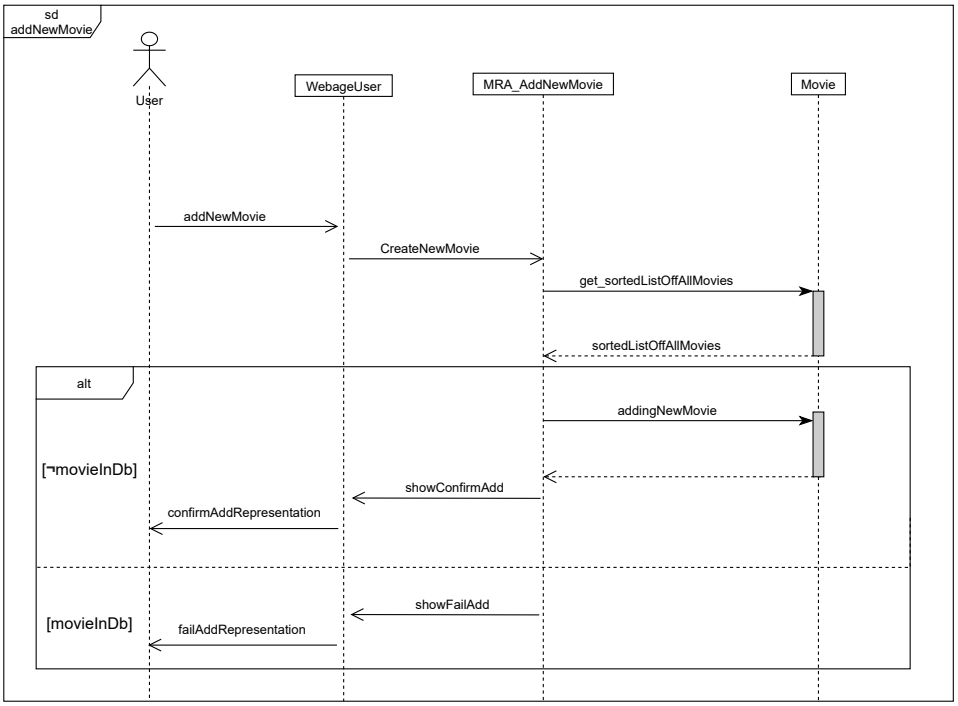

Figure 1.12: SequenceDiagram for R4

R5 Registered Users can access a list of all movies in the database sorted by rating in descending order. For each movie, the average rating is calculated and shown together with the comments.

- Using the domain knowledge

(F3) Each movie can be contained in the database once.

(F2) Each movie has a title, director, at least one character, and original publishing date.

- we can derive the specifications: .

WebpageU(S5a) When the webpage receives the command "viewAllMovies", then the command is forwarded to the machine with the command "getAllMovies". The results are received via the commands "showAllMovies" and "showAvarageRating" and shown to the user by "SortedMoviesRepresentation" and "averageRatingShown".

MRA_ShowMO(S5b) When the machine receives the command "getAllMovies", all movies are brought by the command "get_sortedListOfAllMovies" the rating for each user is brought by "get_movieRating" and the average rating is brought by the command "get_averageRating" and then the data are received as"sortedListOfAllMovies", "movieAverageRating" and "movieRating". The results are returned to the web page via the command "showAllMovies" and "showAverageRating".

Movie(S5c) After receiving the command "get_sortedListOfAllMovies" and "get_averageRating" the results are returned as the data "sortedListOfAllMovies" and "movieAverageRating" .

UserMovieList(S5d) After receiving the command ""get_movieRating" the results are returned as the data "movieRating" .

Correctness condition: $(S5a) \wedge (S5b) \wedge (S5c) \wedge (S5d) \wedge (F3) \wedge (F2) \implies (R5)$

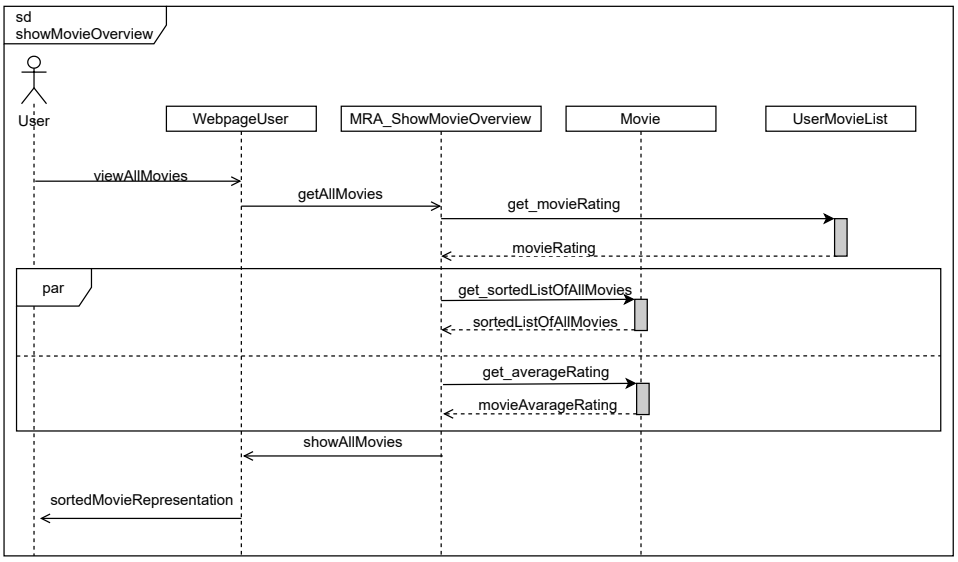

Figure 1.13: SequenceDiagram for R5

1.3.2 Validation

- $S_{abstract} \wedge D$ are non-contradictory.
  No contradictions can be found in $S_{abstract} \wedge D$.

- $S_{abstract} \wedge D \implies R$
  $(S1a) \wedge (S1b) \wedge (S1c) \wedge (F1) \implies (R1)$
  $(S3a) \wedge (S3b) \wedge (S3c) \wedge (F7) \implies (R3)$
  $(S4a) \wedge (S4b) \wedge (S4c) \wedge (F3) \implies (R4)$
  $(S5a) \wedge (S5b) \wedge (S5c) \wedge (S5d) \wedge (F3) \implies (R5)$

- Messages and phenomena are consistent.

message in scenario	source	target	phenomena in problem diagram
register	UnregisteredUser	Webpage-UnregisteredUser	UU!{register}
forwardRegister	Webpage-UnregisteredUser	MRA_Register	WUU!{forwardRegister}
get_userInfo	MRA_Register	UserDatabase	UD!{userInfo}
showFail	MRA_Register	Webpage-UnregisteredUser	MRAR!{showFail}
failAgeRepresentation	Webpage-UnregisteredUser	UnregisteredUser	WUU!{failAgeRepresentation}
registeration	MRA_Register	UserDatabase	MRAR!{registration}
showOk	MRA_Register	Webpage-UnregisteredUser	MRAR!{showOk}
okRepresentation	Webpage-UnregisteredUser	UnregisteredUser	WUU!{okRepresentation}
addNewMovie	User	WebpageUser	U!{addNewMovie}
createNewMovie	WebageUser	MRA_AddNewMovie	WU!{createNewMovie}
get_sortedListOffAllMovies	MRA_AddNewMovie	Movie	M!{sortedListOfAllMovies}
addingNewMovie	MRA_AddNewMovie	Movie	MRAANM!{addingNewMovie}
showConfirmAdd	MRA_AddNewMovie	WebageUser	MRAANM!{showConfirmAdd}
showFailAdd	MRA_AddNewMovie	WebageUser	MRAANM!{showFailAdd}
failAddRepresentation	WebageUser	User	WU!{failAddRepresentation}
confirmAddRepresentation	WebageUser	User	WU!{confirmAddRepresentation}
viewAllMovies	User	WebpageUser	U!{viewAllMovies}
getAllMovies	WebpageUser	MRA_Show-MovieOverview	WU!{getAllMovies}
get_movieRating	MRA_ShowMovie-Overview	UserMovieList	UML!{movieRating}
get_sortedListOfAllMovies	MRA_ShowMovie-Overview	Movie	M!{sortedListOfAllMovies}
showAllMovies	MRA_ShowMovie-Overview	WebpageUser	MRASMO!{showAllMovies}
sortedMovie-Representation	WebpageUser	User	WU!{sortedMovie-Representation}
get_averageRating	MRA_ShowMovie-Overview	Movie	M!{movieAverageRating}
showAverageRating	MRA_Show-MovieOverview	WebpageUser	MRASMO!{showAverageRating}
averageRating-Shown	WebpageUser	User	WU!{averageRating-Shown}
addRating	User	WebpageUser	U!{addRating}

– continued from previous page

message in scenario	source	target	phenomena in problem diagram
forwardRating	WebpageUser	MRA_Rate	WU!{forwardRating}
get_userListOfMovies	MRA_Rate	UserMovieList	UML!{userListOf-Movies}
showRateFail	MRA_Rate	WebpageUser	MRAR!{showRateFail}
rateFailRepresentation	WebpageUser	User	WU!{rateFailRepresentation}
addingRating	MRA_Rate	UserMovieList	MRAR!{addingRating}
showRateOk	MRA_Rate	WebpageUser	MRAR!{showRateOk}
rateOkRepresentation	WebpageUser	User	WU!{rateOkRepresentation}
addOptionalComment	User	WebpageUser	U!{addOptionalComment}
forwardOptionalComment	WebpageUser	MRA_Rate	WU!{forwardOptionalComment}
addingOptionalComment	MRA_Rate	UserMovieList	MRAR!{addingOptional-Comment}
showFeedbackOptional-Comment	MRA_Rate	WebpageUser	MRAR!{showFeedbackOptional-Comment}
feedbackOptional-CommentRepresentation	WebpageUser	User	WU!{feedbackOptional-CommentRepresentation}

- Lexical domains are not sources of messages

message in scenario	source	domain type
register	UnregisteredUser	BiddableDomain
forwardRegister	Webpage-UnregisteredUser	CausalDomain
get_userInfo	MRA_Register	Machine
showFail	MRA_Register	Machine
failAgeRepresentation	Webpage-UnregisteredUser	CausalDomain
registeration	MRA_Register	Machine
showOk	MRA_Register	Machine
okRepresentation	Webpage-UnregisteredUser	CausalDomain
addNewMovie	User	BiddableDomain
createNewMovie	WebageUser	CausalDomain
get_sortedListOffAllMovies	MRA_AddNewMovie	Machine
addingNewMovie	MRA_AddNewMovie	Machine
showConfirmAdd	MRA_AddNewMovie	Machine
showFailAdd	MRA_AddNewMovie	Machine
failAddRepresentation	WebageUser	CausalDomain
confirmAddRepresentation	WebageUser	CausalDomain
viewAllMovies	User	BiddableDomain
getAllMovies	WebpageUser	CausalDomain
get_movieRating	MRA_ShowMovie-Overview	Machine
get_sortedListOfAllMovies	MRA_ShowMovie-Overview	Machine
showAllMovies	MRA_ShowMovie-Overview	Machine
sortedMovieRepresentation	WebpageUser	CausalDomain
get_averageRating	MRA_ShowMovie-Overview	Machine

– continued from previous page

message in scenario	source	domain type
showAverageRating	MRA_Show-MovieOverview	Machine
averageRatingShown	WebpageUser	CausalDomain
addRating	User	BiddableDomain
forwardRating	WebpageUser	CausalDomain
get_userListOf-Movies	MRA_Rate	Machine
showRateFail	MRA_Rate	Machine
rateFailRepresentation	WebpageUser	CausalDomain
addingRating	MRA_Rate	Machine
showRateOk	MRA_Rate	Machine
rateOkRepresentation	WebpageUser	CausalDomain
addOptionalComment	User	BiddableDomain
forwardOptionalComment	WebpageUser	CausalDomain
addingOptionalComment	MRA_Rate	Machine
showFeedbackOptional-Comment	MRA_Rate	Machine
feedbackOptionalCommentRepresentation	WebpageUser	CausalDomain

- There exists at least one scenario for each subproblem.
- Scenarios cover normal cases and possibly exceptional cases.

subproblem	normal case	exceptional case
pdRegister	sdRegister	sdRegister
pdRate	sdRate	sdRate
pdAddNewMovie	sdAddNewMoive	
pdShowMovieOverview	sdShowMovieOverview	

Only for pdRegister & pdRate an exceptional case must be considered.

1.4 A4

1.4.1 Technical Software Specification

Technical realization of domains from context and problem diagrams

- Movie, UserMovieList, UserDatabase : Realized as SQLDatabase on the same computer as the machine. Therefore, the database is connected by a cal- and- return interface and used SQL commands. We have combine our whole Lexical-designed domain in one.
- WebpageUnregisteredUser: Realized using Apache Tomcat and UnregisteredWebBrowser(Browser of Unregistered User).
- WebpageUser:Realized using ApacheTomcat and WebBrowser User (browser of Use)) *We decide to use ApacheTomcat as a server platform, because the customer realized other projects on this platform and requires a Java implementation.*

Technical Contextdiagram & Mapping

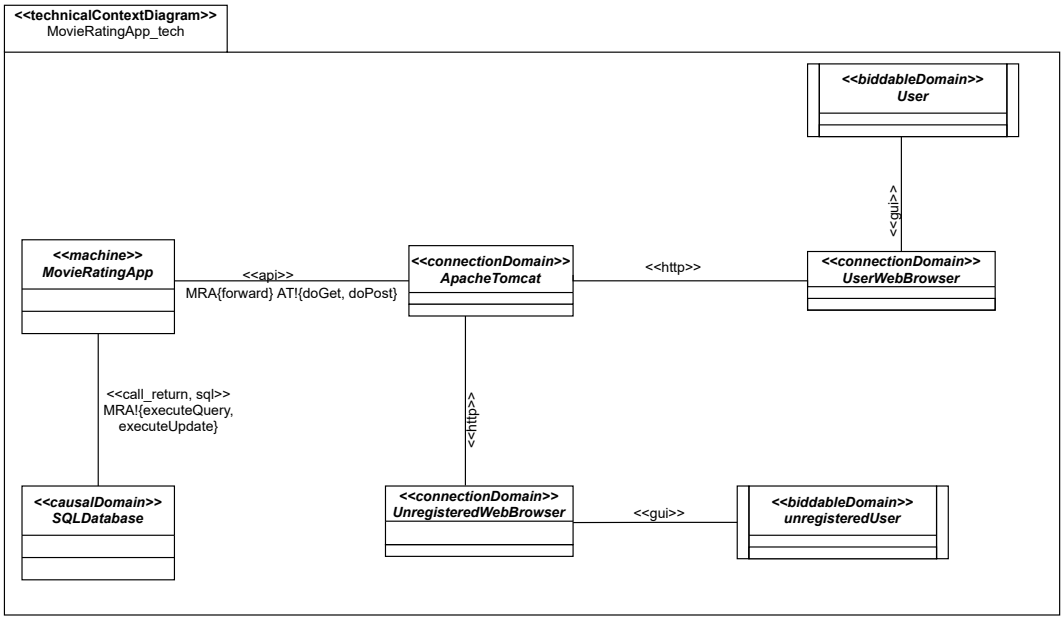

Figure 1.14: Technical Contextdiagram

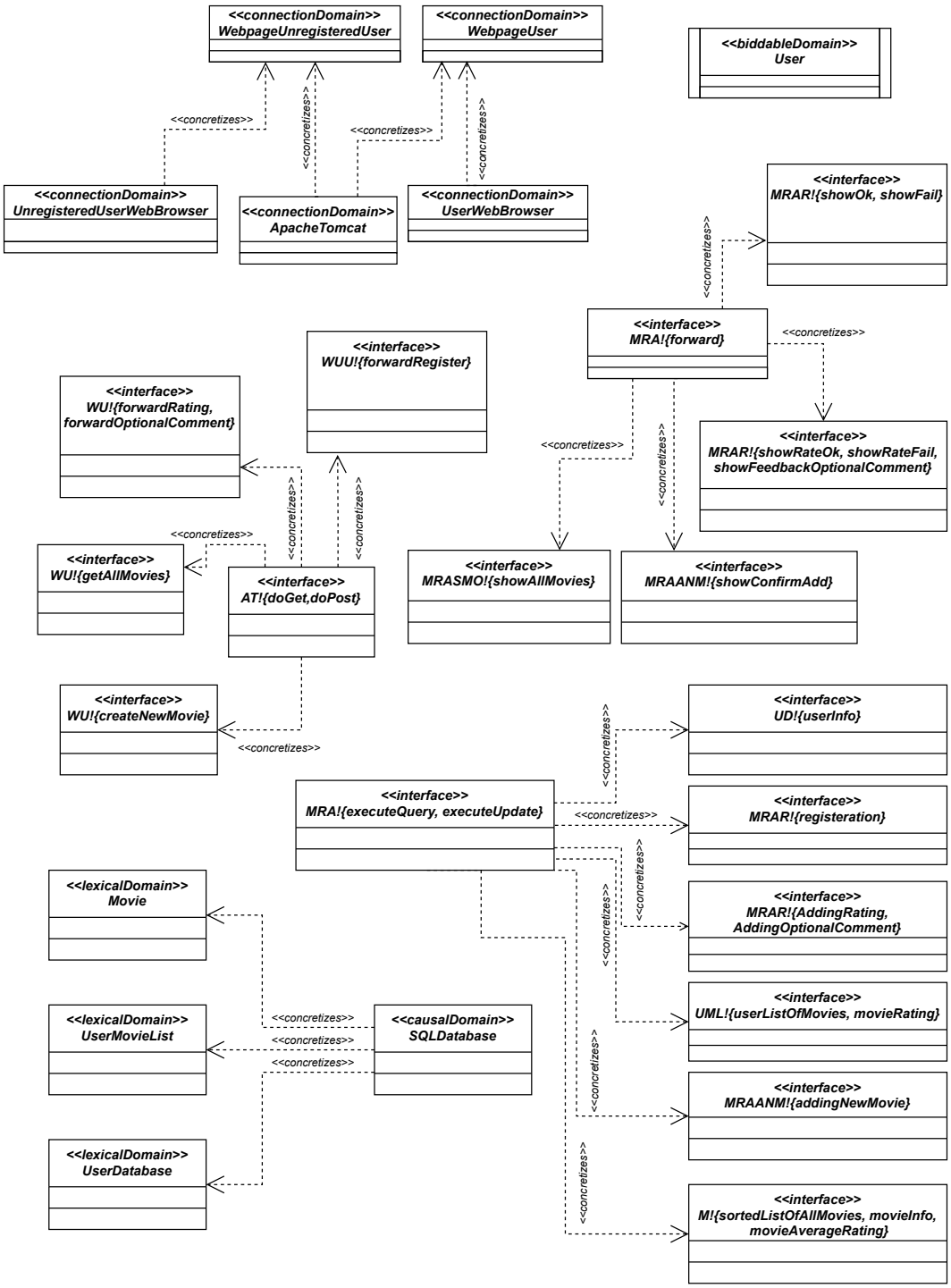

Figure 1.15: Mapping diagram for Technical Contextdiagram

1.4.2 Validation

- New phenomena and domains are suitable to implement the external messages used in the abstract phenomena:

Message	new phenomena and Domains
register	ApacheTomcat, HTTP
addRating , addOptionlaComment	ApacheTomcat, HTTP
addNewMovie	ApacheTomcat, HTTP
viewAllMovies	ApacheTomcat, HTTP

- All internal messages can be realized using SQL commands.
- All domains of the technical context diagram are related to domains in the problem diagrams:
- All phenomena in the technical context diagram are related to elements in the problem diagrams:

\+ *Provided mapping diagram*

- All domains directly connected with the machine in the problem diagrams are related to elements in the technical context diagram:

Problem Diagram	Domain Connected with the machine	Element in TCD
Register	UserDatabase	SQLDatabase
	WebpageUnregisteredUser	UnregisteredUserWebbrowser, APacheTomcat
Rate	UserMovieList	SQLDatabase
	WebpageUser	UserWebbrowser, APacheTomcat
AddNewMovie	Movie	SQLDatabase
	WebpageUser	UserWebbrowser, APacheTomcat
showMovieOverview	UserMovieList	SQLDatabase
	WebpageUser	UserWebbrowser, APacheTomcat

1.5 A5

1.5.1 Class Model and Operations Specification

forwardRegister

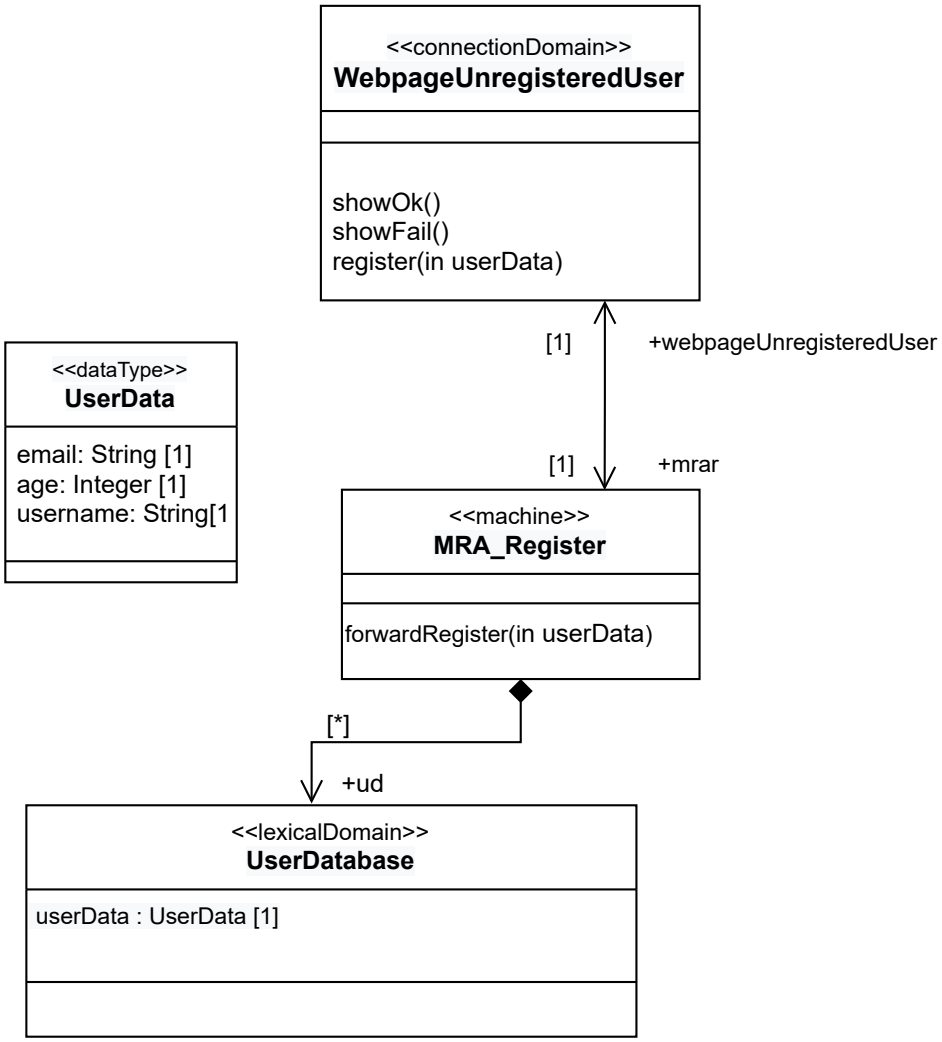

Figure 1.16: Class Model

Name: forwardRegister

Descripotion:forward the registeration request and return confirm or fail

OCL constraint: forwardRegister

context MRA_Register::forwardRegister(userData: UserData)
pre: true
post: if userData.age >= 18 and not ud – > one (u: userDatabase|userData.username = u.userData.username) then
ud->one(u:UserDatabase | u.UserData = userData) and ud->size() = ud @pre->size()+1 and
webpageUnregisteredUserˆshowOk()
else webpageUnregisteredUserˆshowFail()
end if

OCL constraint:

context UserDatabase
inv: UserDatabase.allInstances()->isUnique(userData.username)

forwardRegister Validation

- Operation specifications must be consistent with abstract specifications:
  The Operation specification of forwardRegister is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition.
- Parameters must be used in the pre- and/or postcondition: *The parameters are used in the postcondition.*
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  To register User can input all parameters to the WebpageUnregisteredUser via his/her web browser, which forwards these to this operation.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  A new class RegisterationData is introduced to represent a new registration request. It has an association with the class UserDatabase named BelongsTo. The attribute RegisterationDate: TimeData is added to the class RegisterationData.

createNewMovie

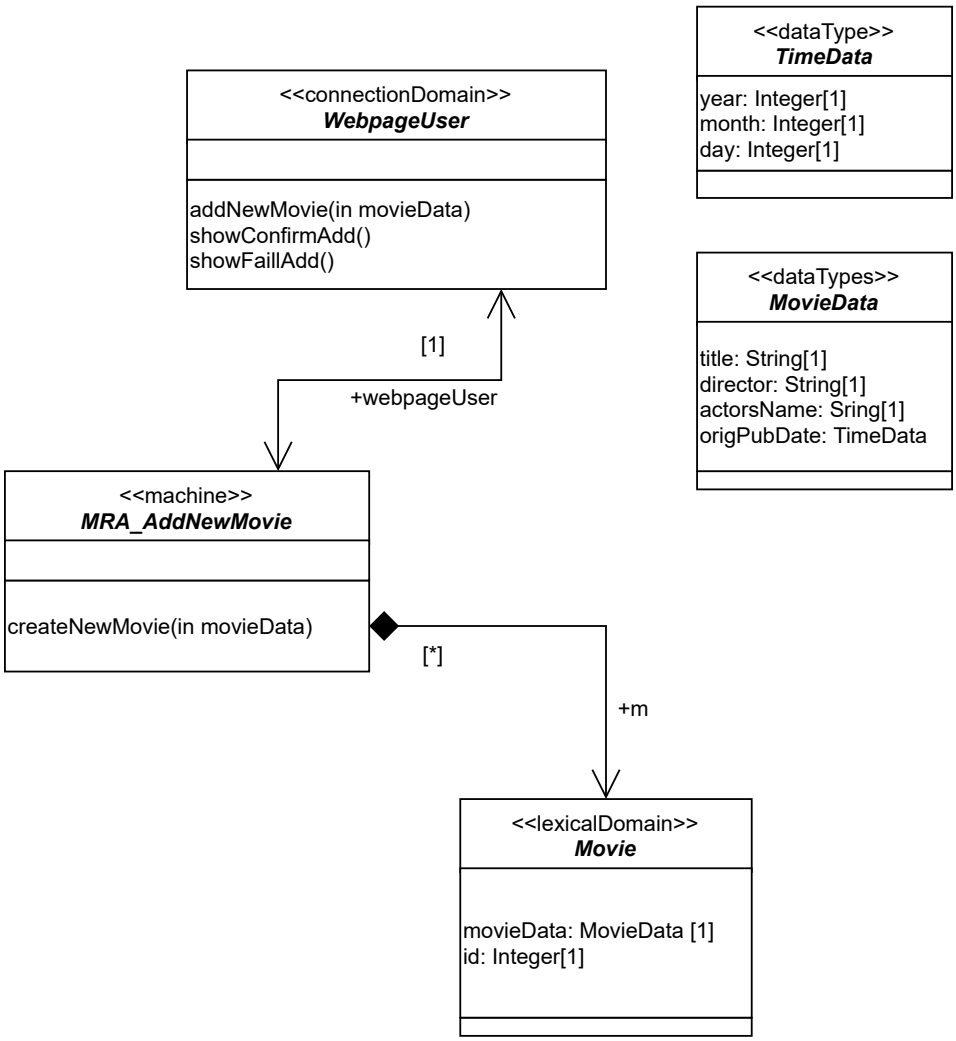

Figure 1.17: Class Model

Name: createNewMovie

Descripotion: creates a new movie and return fail or confirm

OCL constraint:

context MRA_AddNewMovie::createNewMovie(movieData: MovieData)
pre: true
post: if not mo ->one (m: Movie | movieData = m.movieData) then
m -> one(md:MovieData | md.movieData = movieData) and m -> size() = m@pre -> size()
+1 and
webpageUserˆshowConfirmAdd() else
webpageUserˆshowFailAdd()
end if

OCL constraint:

context Movie
inv: Movie.allInstances()isUnique->(movieData)

createNewMovie Validation

- Operation specifications must be consistent with abstract specifications:
  The Operation specification of createNewMovie is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition.
- Parameters must be used in the pre- and/or postcondition: *The parameters are used in the postcondition.*
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  To add a new movie, User can input all parameters to the WebpageUser via his/her web browser, which forwards these to this operation.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  A new class adding is introduced to represent a new movie adding request. It has an association with the class Movie named BelongsTo.

showAllMovies

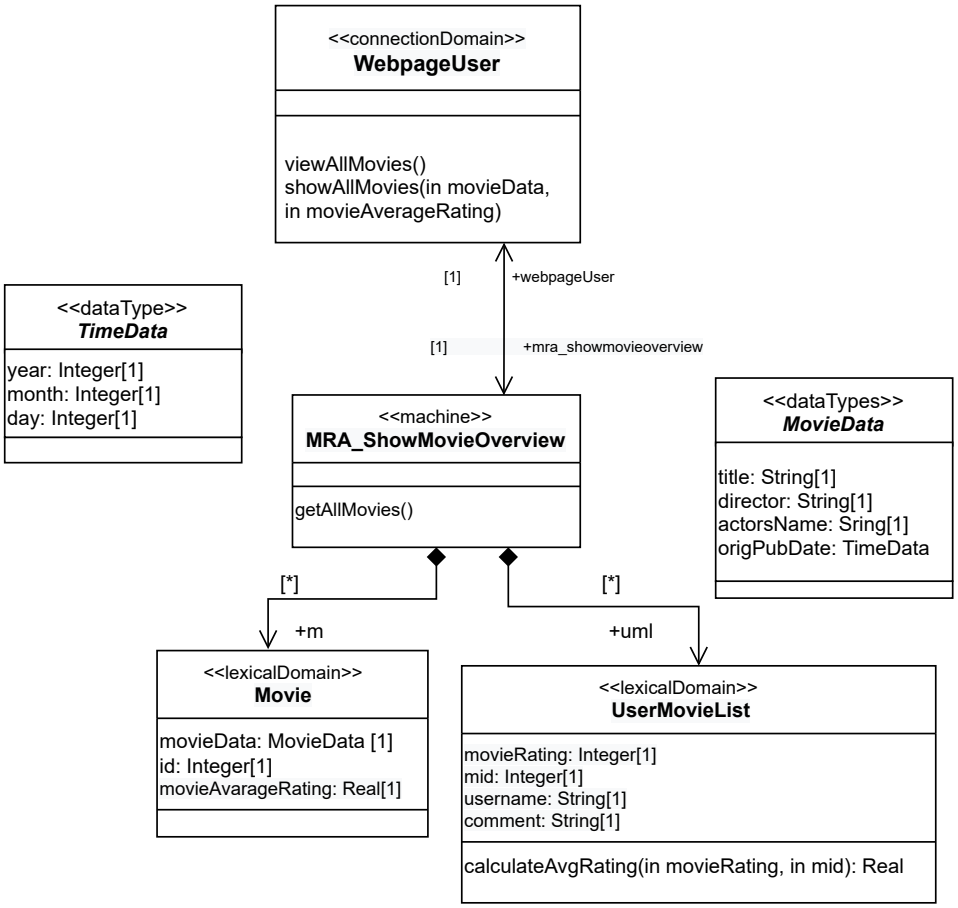

Figure 1.18: Class Model

Name: getAllMovies

Descripotion: Forwards the view request from the User to the machine.

OCL constraint:

context WebpageUser::getAllMovies()
pre: true
post:m:Movie.movieAvarageRating=uml:UserMovieListˆcalculateAvgRating (uml.movieRating,m.mid) let res: OrderedSet(Movie) = m->sortedBy(m:Movie | m.movieAvarageRating)
in webpageUserˆshowAllMovies(res)

OCL constraint:

context Movie
inv: Movie.allInstances()isUnique->(movieData)

showAllMovies Validation

- Operation specifications must be consistent with abstract specifications:

The Operation specification of showAllMovies is consistent with the abstract specification.

- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition.
- Parameters must be used in the pre- and/or postcondition: *The parameters are used in the postcondition.*
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  To show a sorted list of all movies, User can make a request to the WebpageUser via his/her web browser, which forwards this to this operation. The machine knows the argument res used in the message to WebpageGuest.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  No new methods are added.

Rate

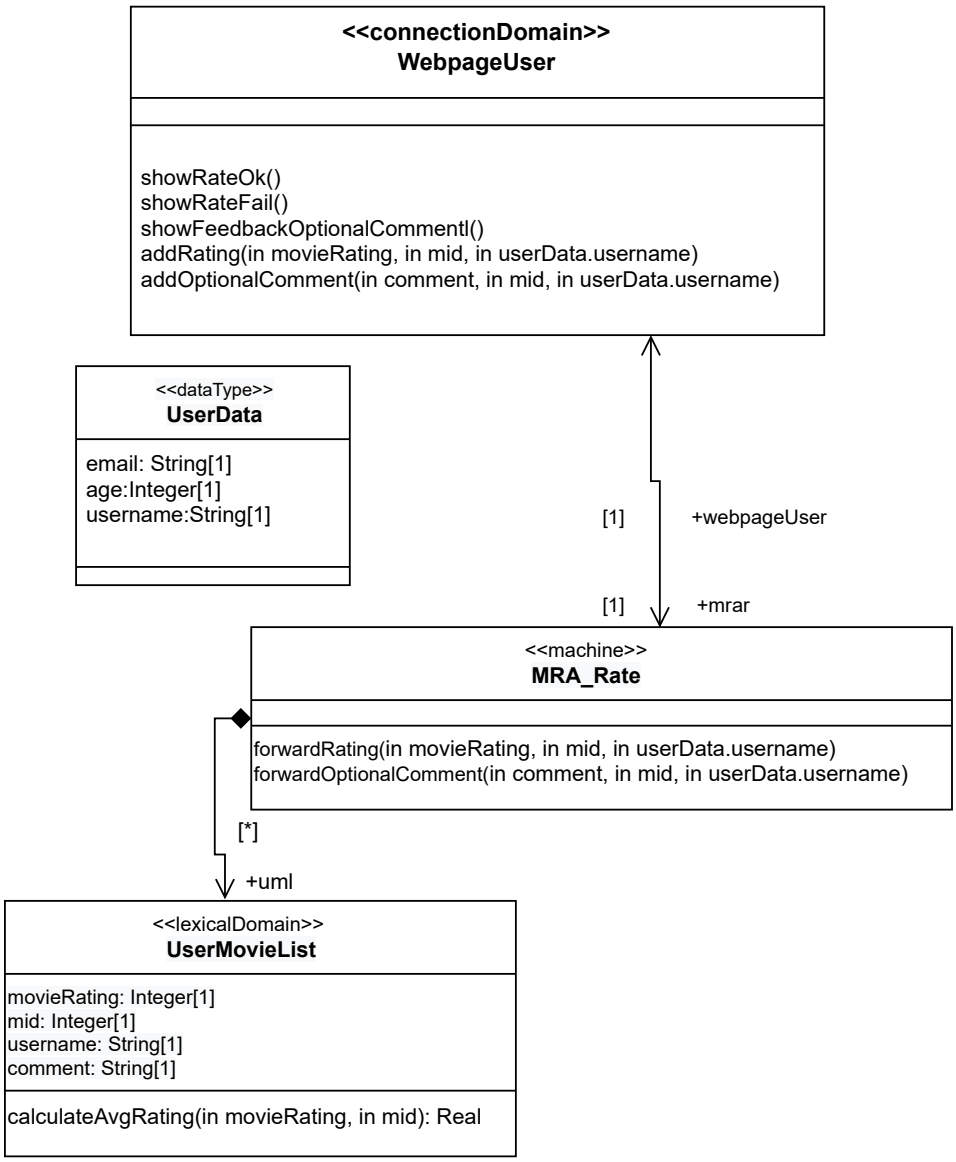

Figure 1.19: Class Model

Name: forwardRating

Descripotion: Forwards the rate request from the User to the machine.

OCL constraint:

context MRA_Rate::forwardRating(movieRating::Float,mid::Integer,username::string)
pre: not uml -> exist (u: UserMovieList | mid = u.mid AND userData.username = u.username)
post: if 0 <= movieRating AND movieRating <= 10 then
uml -> any (u: UserMovieList | mid = u.mid AND
u.movieRating = movieRating AND
u.username = userData.username) AND
uml->size() = uml @pre->size() +1 AND
webpageUserˆShowRateOk()
else webpageUserˆShowFailOk()
end if

Name: forwardOptionalComment

Descripotion: Forwards the comment request from the User to the machine.

OCL constraint:

contextMRA_Rate::forwardOptionalComment(comment::String,mid::Integer, userData.username::string)
pre: not uml -> exist (u: UserMovieList | mid = u.mid AND userData.username = u.username)
post: if comment not = "" then
uml -> one(u: UserMovieList| mid = u.mid
u.comment = comment) AND
webpageUserˆShowFeedbackOptionComment()
end if

OCL constraint:

context UserMovieList
inv: UserMovieList.allInstances()->isUnique(id)

Rate Validation

- Operation specifications must be consistent with abstract specifications:
  The Operation specification of forwardRating and forwardOptionalComment is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition..
- Parameters must be used in the pre- and/or postcondition: *The parameters are used in the postcondition.*
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  User can add ratings,User can input all parameters to the WebpageUser via his/her web browser, which forwards these to this operation.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  no new methods are added.

State predicate definitions

State predicate	Parameters	Definition
RatingAdded	id, signature	uml -> exist (u: UserMovieList | id = u.id AND signature = u.signature)

1.6 A6

1.6.1 Software Lifecycle

- $LC_{unregisteredUser} = (Register)^+$
- $LC_{User} = (AddNewMovie|ShowMoviesOverview|AddRating;[AddOptionalComment])^*$
- $LC_{MovieRating} = (||_{i=1}^{n} LC_{unregisteredUser})||(||_{i=1}^{m} LC_{User})$

  where $||_{i=1}^{n} LC$ denotes the parallel composition of n copies of life-cycle LC
  and $||_{i=1}^{m} LC_{User})$ denotes the parallel composition of m copies of life-cycle LC

1.6.2 Validation

- Each sequence diagram of Step A3: Abstract software specification is contained in at least one life-cycle expression.

Scenario	life-cycle expression
sdRegister	$LC_{unregisteredUser}$
sdRate	LC_{User}
sdAddNewMovie	LC_{User}
sdShowMovieOverview	LC_{User}

- For each biddable domain exists exactly one life-cycle.

biddable domain	life-cycle expression
UnregisteredUser	$LC_{unregisteredUser}$
User	LC_{User}

- The life-cycles are consistent with the state predicates in Step A3: Abstract software specification:
  - Register has no state predicates at the beginning and end. Hence, it must be exactly one time for the user to use app functionality.
  - addRating can be executed if a Rating object is not created beforehand. Otherwise, Rate return an error.
  - AddOptionalComment can be executed if a Rating object is created beforehand.
  - ShowMovieOverview has no state predicates at the beginning. Hence, it can be executed an arbitrary number of times.
- the life-cycles are consistent with the pre- and postconditions in Step A5: Operations and data specification:
  - The sequence diagram Register contains the operation forwardRegister. forwardRegister requires, that a user with the same username does not exist . This is ensured by the precondition.
  - The sequence diagram ShowMovieOverview contains the operation getAllMovies. It has no precondition. Hence, it can be executed at any position of the life-cycle.
  - The sequence diagram Rate contains the operation forwardRating. forwardRating requires, that a rating from the same user does not exist. this is ensured in the postcondition
  - The sequence diagram Rate contains the operation forwardOptionalcomment. forwardOptionalcomment requires, that a rating from the same user does exist . This is ensured by the precondition.
- Exactly one life-cycle exists for the machine domain, that combines all life-cycles.
  The life-cycle $LC_{MovieRating}$ exists for the machine domain. It combines all life-cycles.

2 Design

2.1 D1

2.1.1 Software architecture

- MRA_Register fits to update (2)

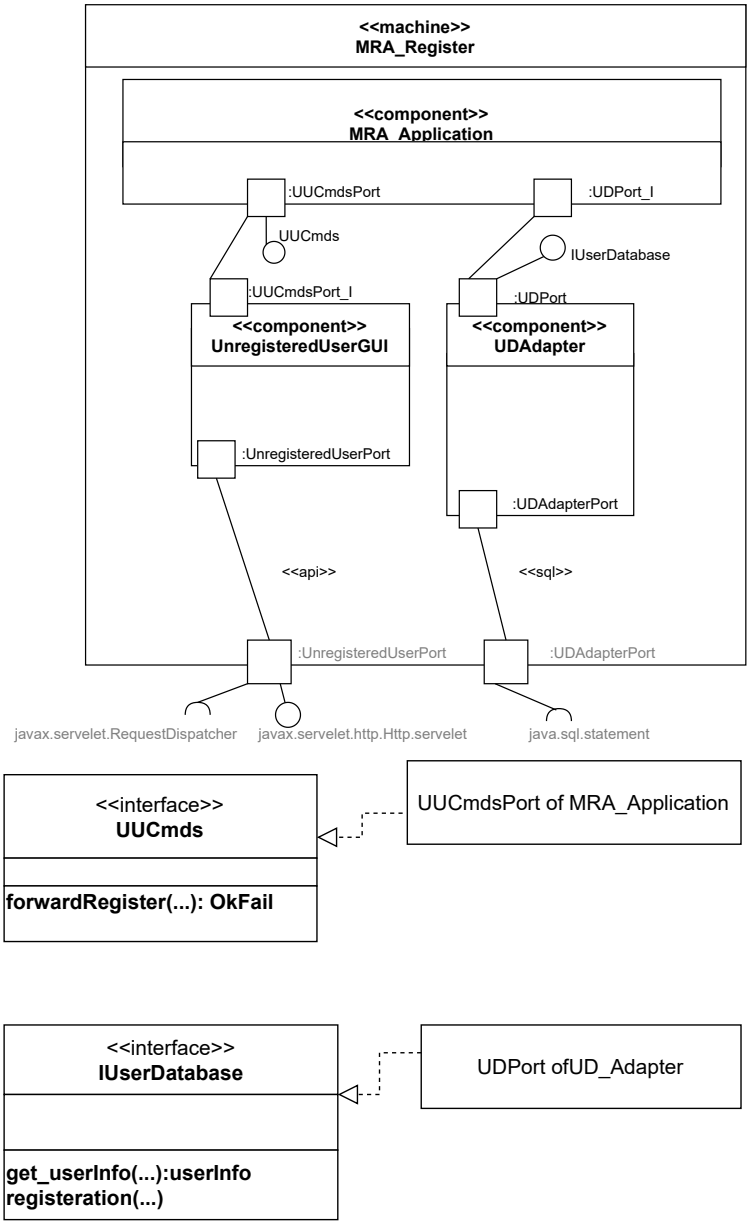

Figure 2.1: subproblem architectures and Internal Interfaces

- Port types and interface relations for MRA_Register

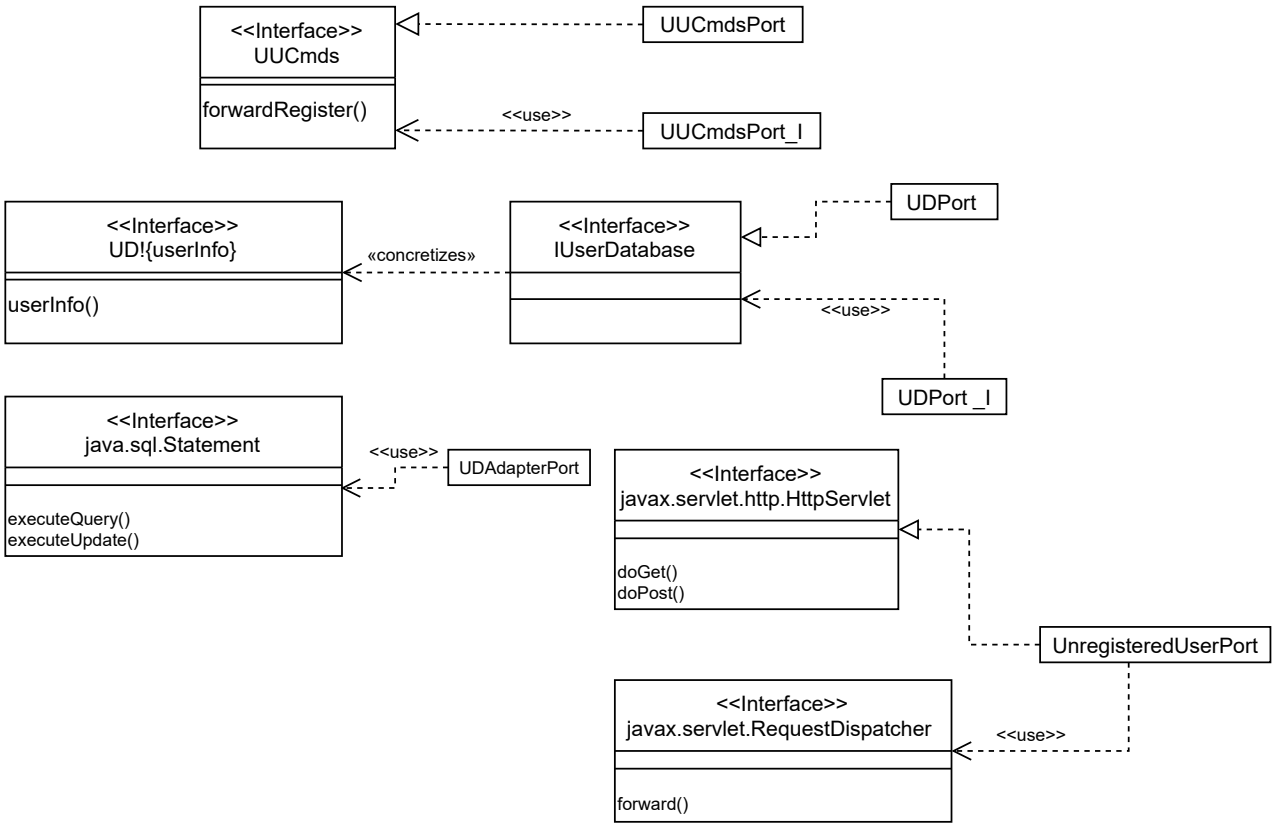

Figure 2.2: Port types and interface relations

- MRA_Rate fits to update (2)

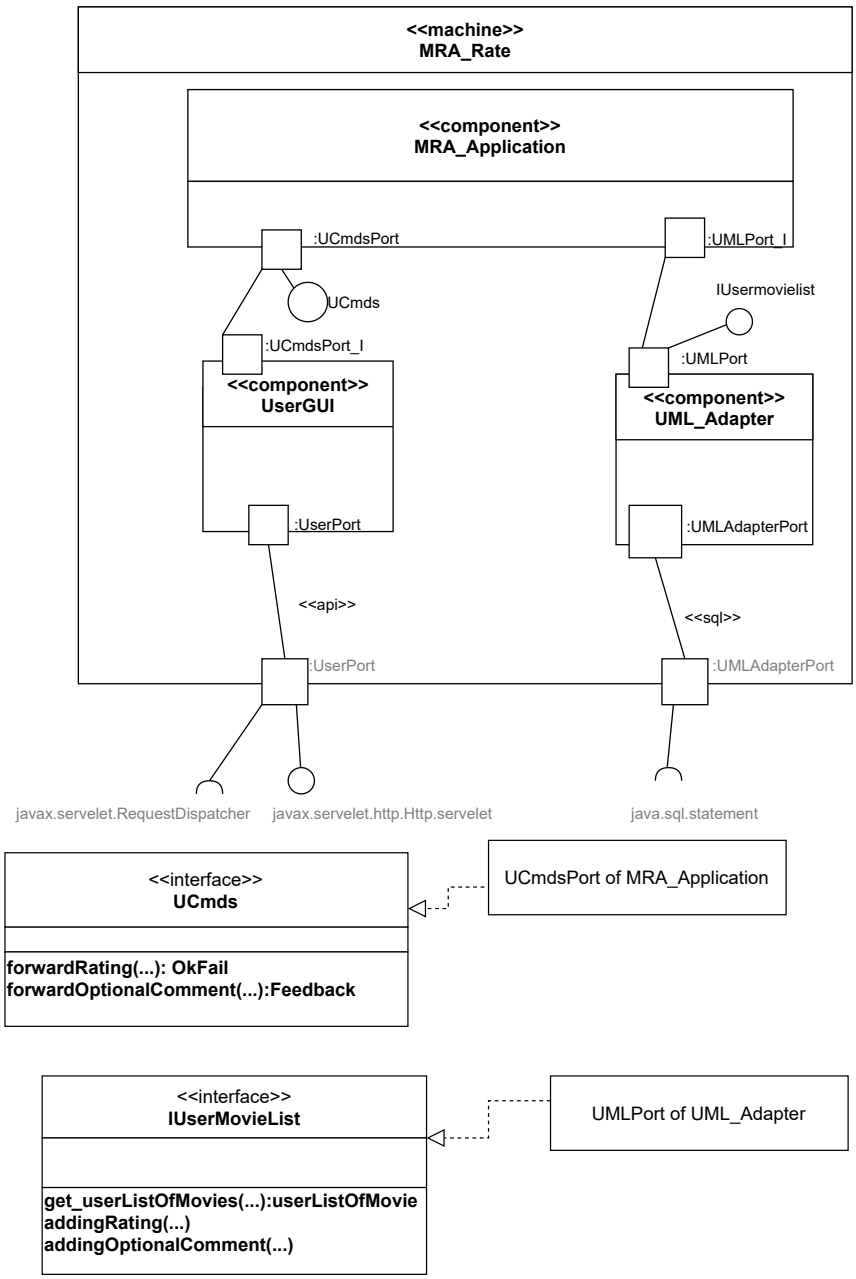

Figure 2.3: subproblem architectures and Internal Interfaces

- Port types and interface relations for MRA_Rate

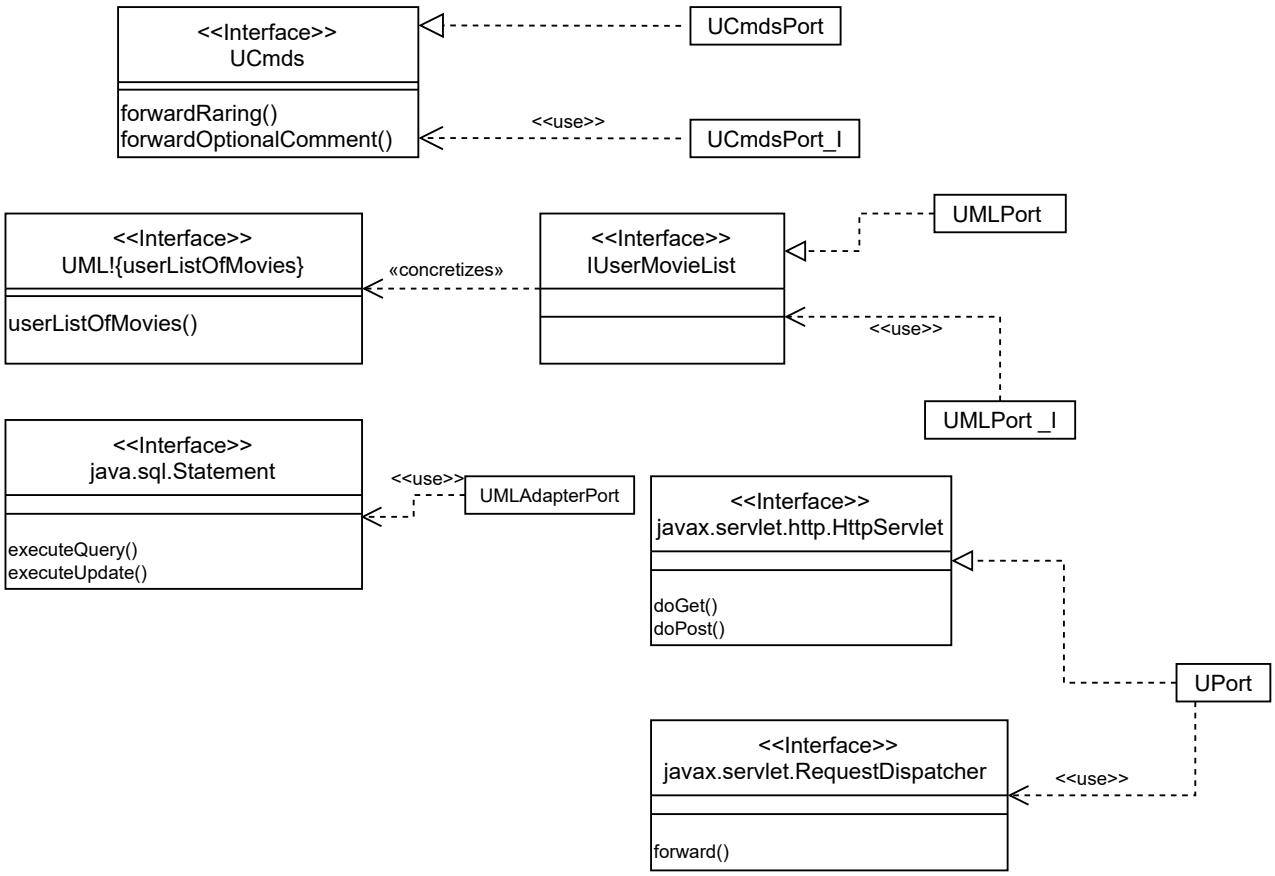

Figure 2.4: Port types and interface relations

- MRA_AddNewMovie fits to update (2)

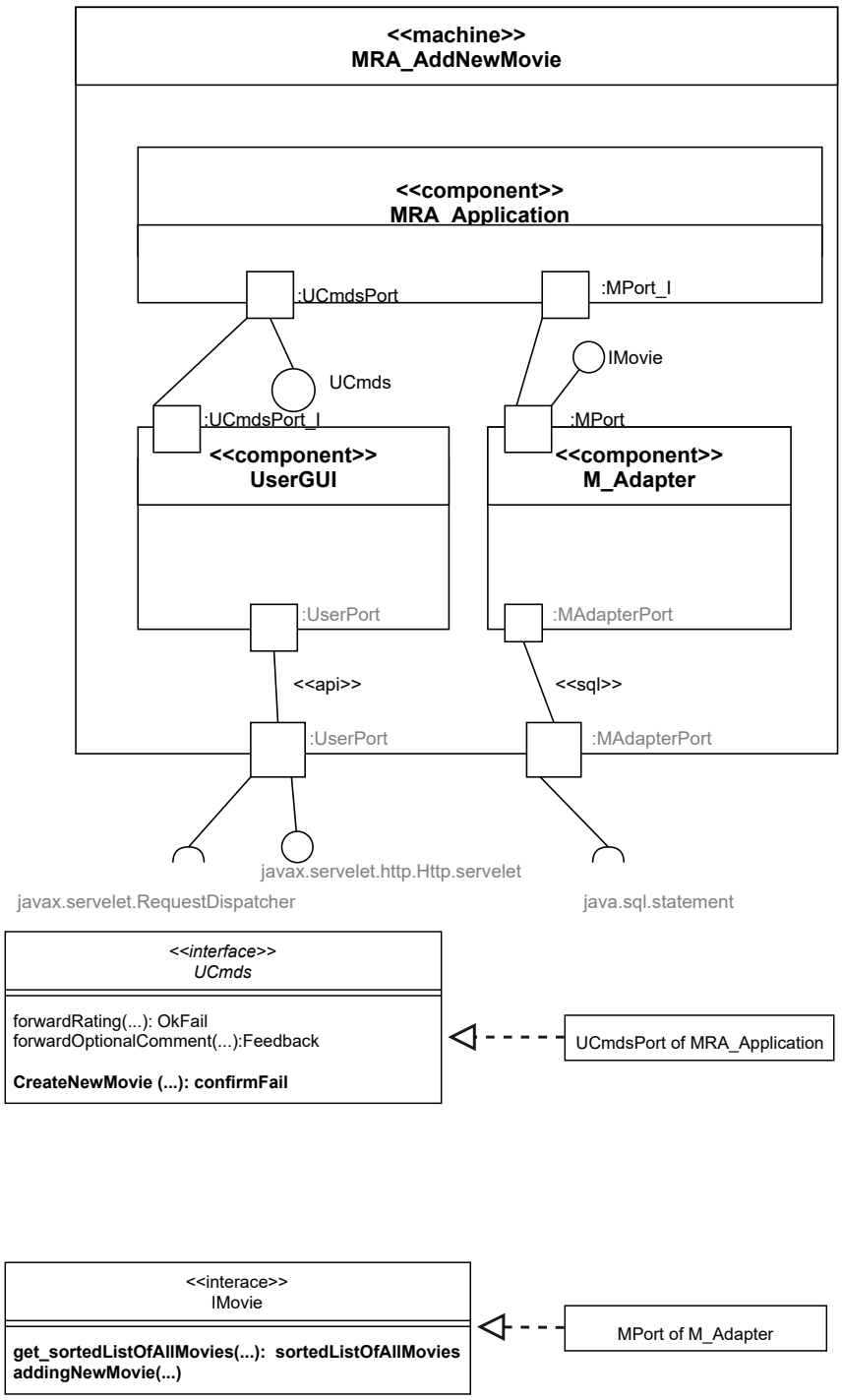

Figure 2.5: subproblem architectures and Internal Interfaces

- Port types and interface relations for MRA_AddNewMovie

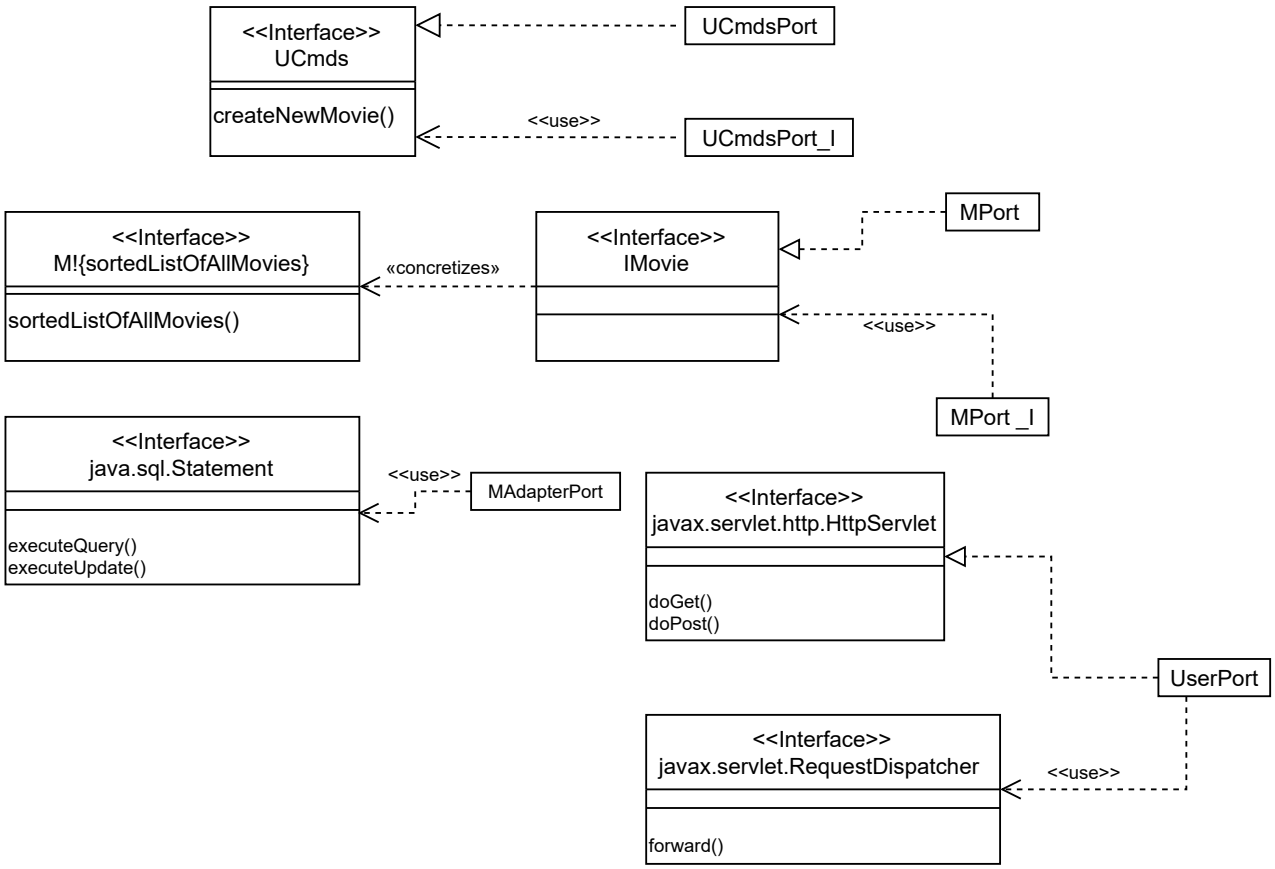

Figure 2.6: Port types and interface relations

- MRA_ShowMovieOverview fits to query (2) variant

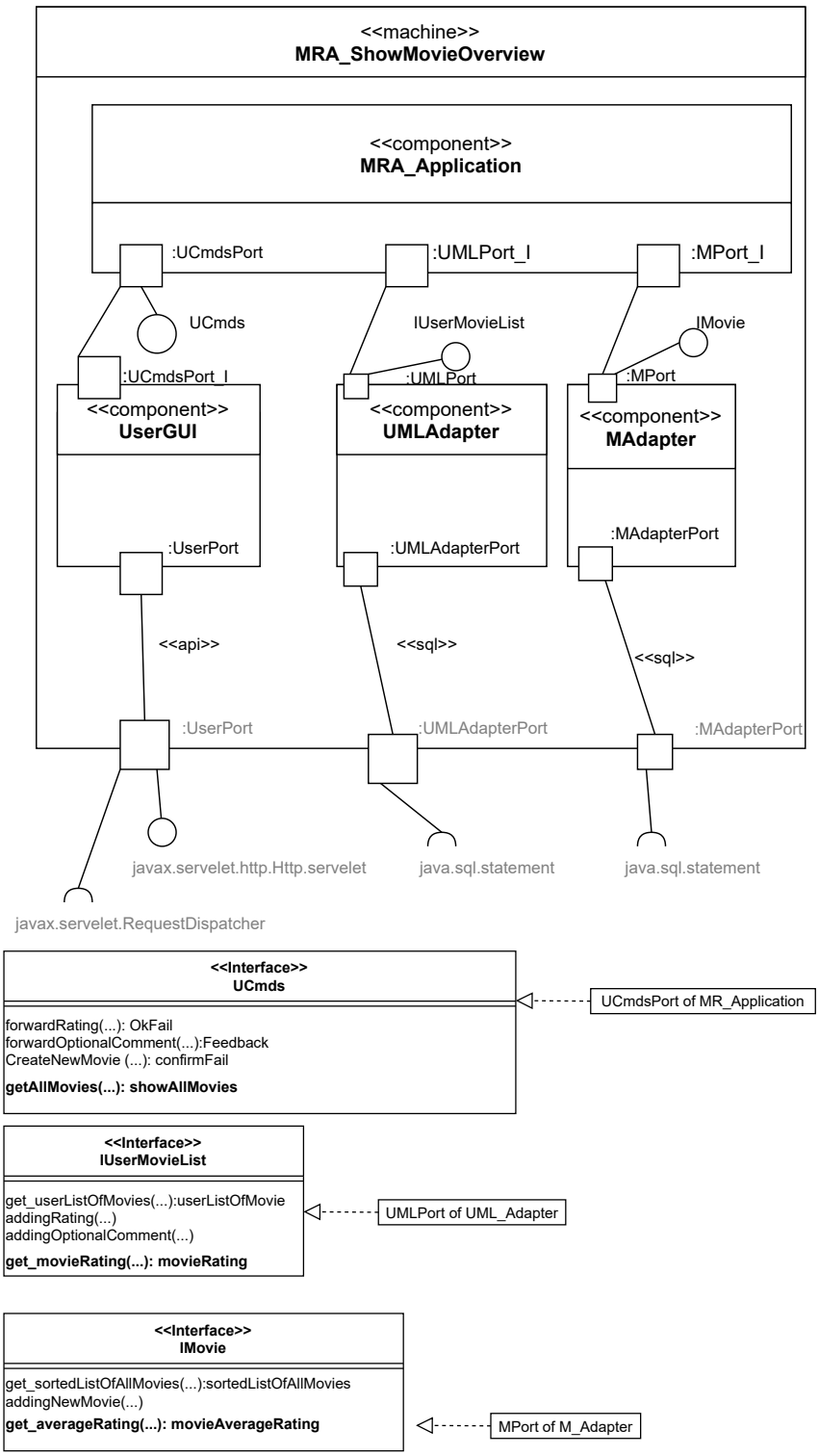

Figure 2.7: subproblem architectures and Internal Interfaces

- Port types and interface relations for MRA_ShowMovieOverview

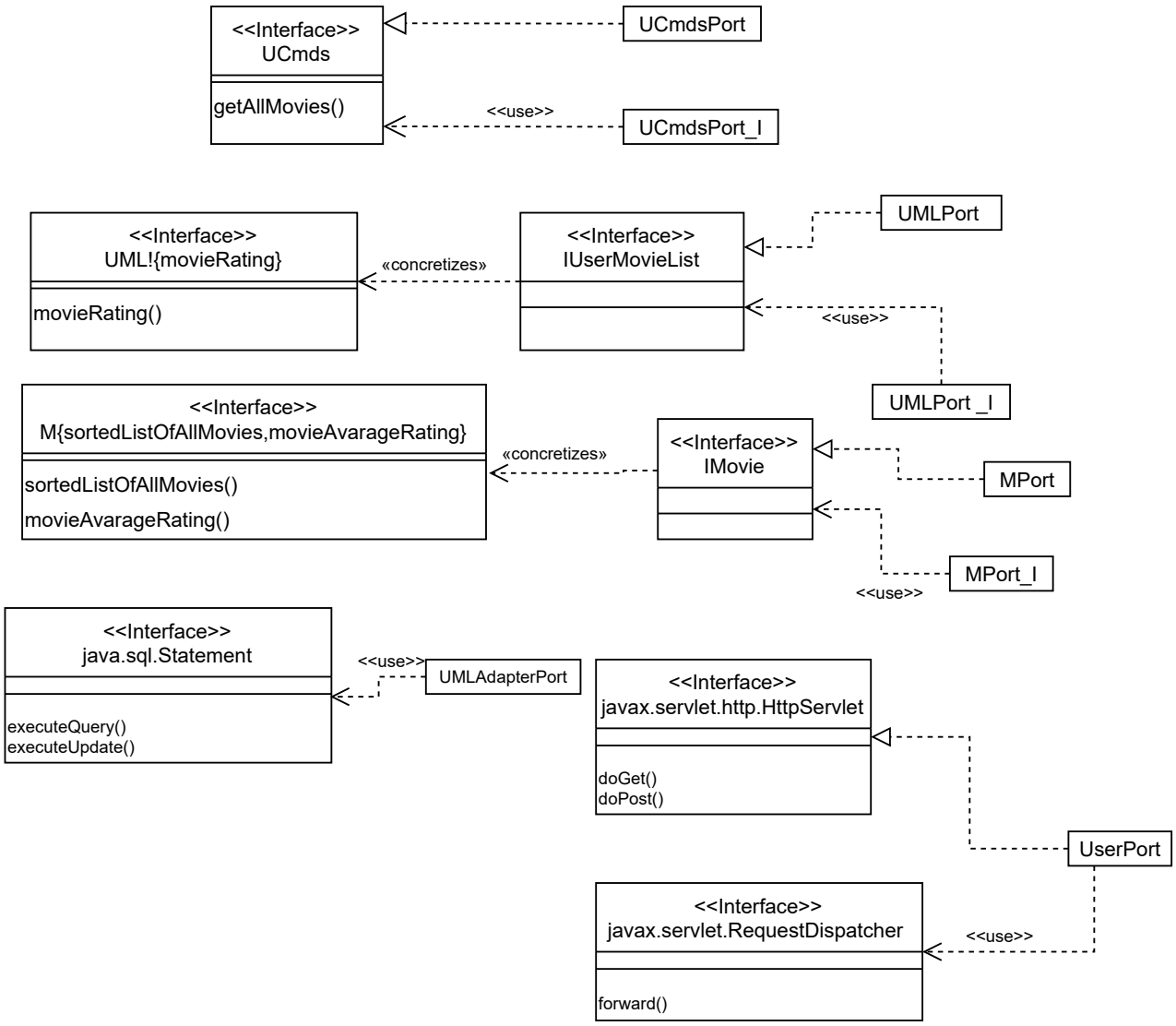

Figure 2.8: Port types and interface relations

2.1.2 Refining app_if interface classes

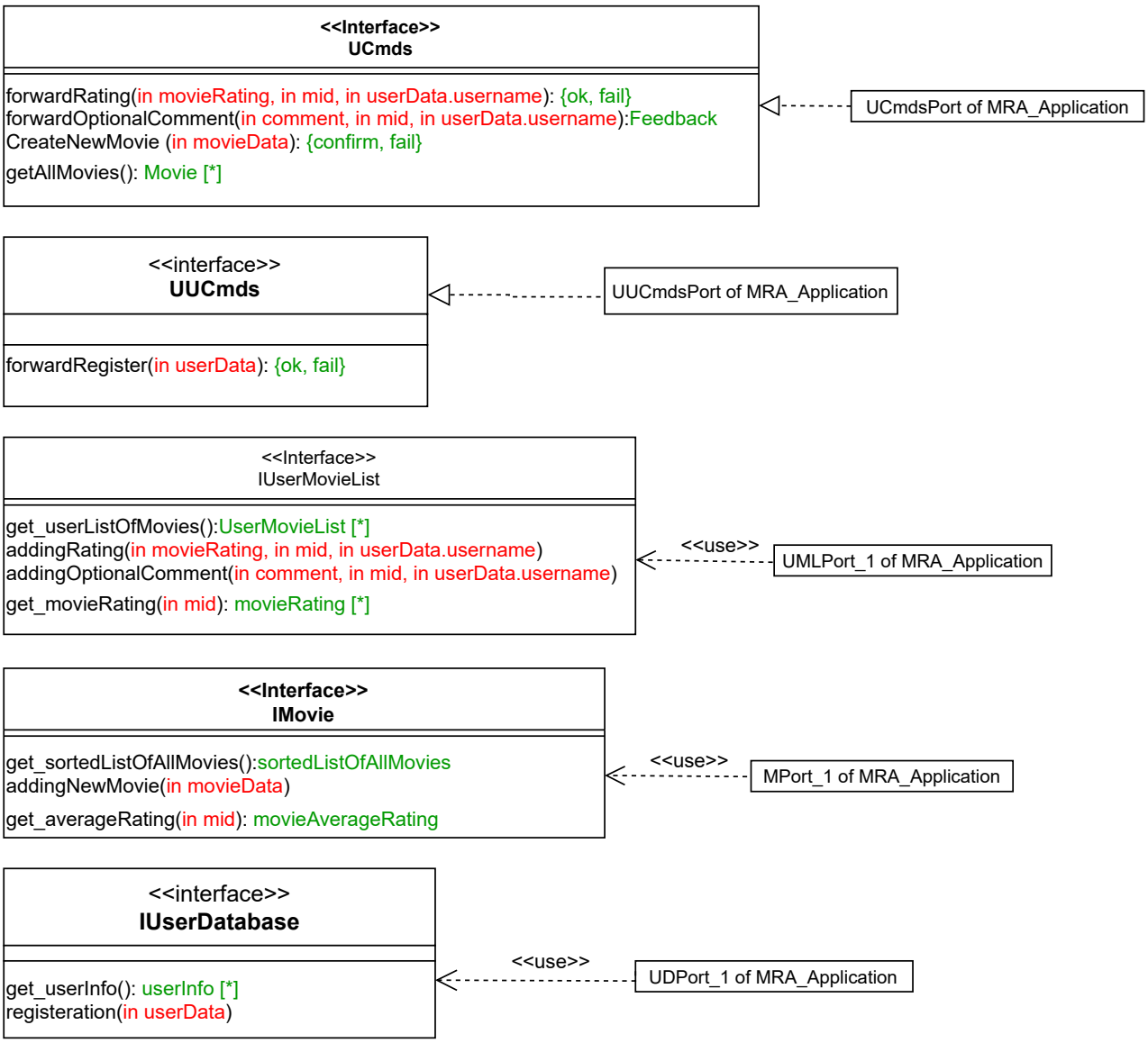

Figure 2.9: Refining app_if interface classes

2.1.3 Refining tech_if interface classes

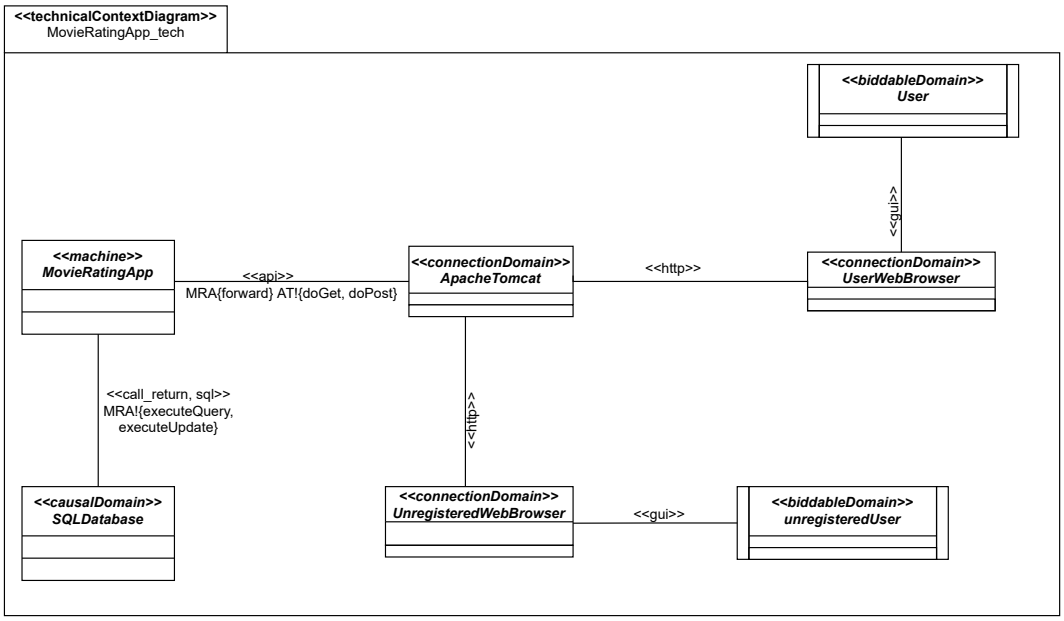

Figure 2.10: Technical Contextdiagram

Considered interface in subproblem architecture	**technical interface**
<<api>>javax.servlet.http.HttpServlet in MRA_Register	<<api>> AT!{doGet, doPost}
<<api>>javax.servlet.RequestDispatcher in MRA_Register	MRA{forward}
<<api>>java.sql.Statement in MRA_Register	<<call_return,sql>> MRA!{executeQuery,executeUpdate}
<<api>>javax.servlet.http.HttpServlet in MRA_Rate	<<api>> AT!{doGet, doPost}
<<api>>javax.servlet.RequestDispatcher in MRA_Rate	MRA{forward}
<<api>>java.sql.Statement in MRA_Rate	<<call_return,sql>> MRA!{executeQuery,executeUpdate}
<<api>>javax.servlet.http.HttpServlet in MRA_AddNewMovie	<<api>> AT!{doGet, doPost}
<<api>>javax.servlet.RequestDispatcher in MRA_AddNewMovie	MRA{forward}
<<api>>java.sql.Statement in MRA_AddNewMovie	<<call_return,sql>> MRA!{executeQuery,executeUpdate}
<<api>>javax.servlet.http.HttpServlet in MRA_ShowMovieOverview	<<api>> AT!{doGet, doPost}
<<api>>javax.servlet.RequestDispatcher in MRA_ShowMovieOverview	MRA{forward}
<<api>>java.sql.Statement in MRA_ShowMovieOverview	<<call_return,sql>> MRA!{executeQuery,executeUpdate}

2.1.4 Refining adapter_if interface classes

- There are no HAL components in the subproblem architectures. Hence, there are no adapter_if interface classes that need to be refined

2.1.5 Merged Architecture

- The components can be merged a follows:
  - The application components in all architectures for the subproblems should be merged because the Subproblems Rate, AddNewMovie and ShowMovieOverview are related sequentially. The Subproblem Register is also merged because of reasons of simplicity
  - The UD Adapters, UML Adapters and M Adapters in all architectures for the subproblems should be merged because it is an adapter, establishing the connection to the DB.
  - The UserGUI for the User and The UnrigesteredUserGUI for the UnrigesteredUser in all architectures for the subproblems uses the same technology and should be merged.
- Components to Develop
  - We have to develop the MRA Application, a UserGUI, a UnregisteredUserGUI, and adapters for the external components.
  - The UD Adapter, UML Adapter and M Adapter are responsible to create and maintain tables for all persistent classes.

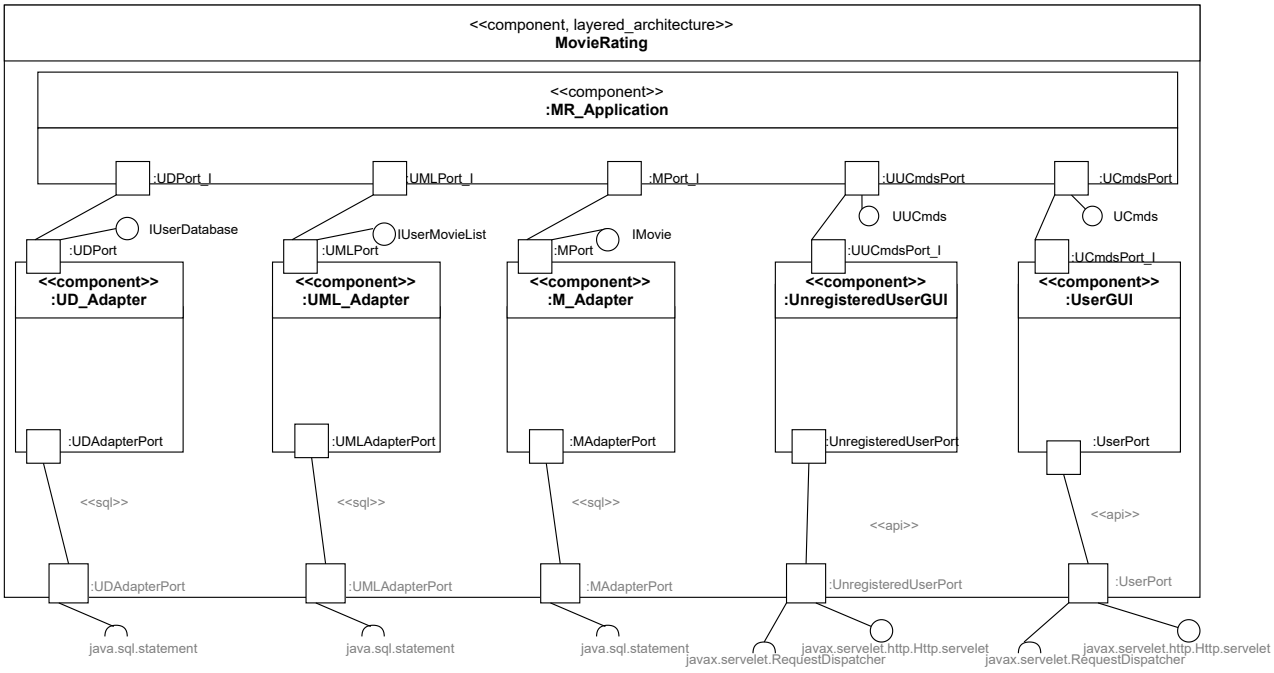

Figure 2.11: Merged Architecture

2.1.6 Validation

- Messages of Step A3: Abstract software specification sequence diagrams are interfaces of the application layer.

Sequence Diagram	Message	in/ out	Application Layer Interface	required/ provided
Register	forwardRegister()	in	UUCmds::forwardRegister()	provided
	get_Userinfo()	out	IUserDatabase::get_Userinfo()	required
	userinfo()	in	return value of IUserDatabase::get_Userinfo()	required
	showFail()	out	return value of UUCmds::forwardRegister()	provided
	showOk()	out	return value of UUCmds::forwardRegister()	provided
Rate	forwardRating()	in	UCmds::forwardRating()	provided
	get_UserListOfMovies()	out	IUserDatabase::get_User ListOfMovies()	required
	userListOfMovies()	in	return value of IUserDatabase::get_User ListOfMovies()	required
	showRateFail()	out	return value of UCmds::forwardRating()	provided
	showRateOk()	out	return value of UCmds::forwardRating()	provided
AddNewMovie	CreateNewMovie()	in	UCmds::CreateNewMovie()	provided
	get_Usersorted ListOffAllMovies()	out	IMovie::get_Usersorted ListOffAllMovies()	required
	sortedListOfAllMovies()	in	return value of IMovie ::get_ UsersortedListOffAllMovies()	required
	addingNewMovie()	out	IMovie::addingNewMovie()	required
	showConfirmAdd()	out	return value of UCmds::forwardRating()	provided
	showFailAdd()	out	return value of UCmds::forwardRating()	provided
showMovieOverview	getAllMovies()	in	UCmds::getAllMovies()	provided
	get_movieRating()	out	IUserMovieList:: get_movieRating()	required
	get_sorted ListOfAllMovies()	out	IMovie:: get_sortedListOfAllMovies()	required
	sorted ListOfAllMovies()	in	IMovie::sorted ListOfAllMovies()	required
	showAllMovies()	out	UCmds::showAllMovies()	provided

- For global architecture: direction of all messages consistent to each other and input.

provided by machine	required by adapter / provided by app
javax.servlet.http.HttpServlet	UCmds / UUCmds

required by machine	provided by adapter / required by app
javax.servlet.RequestDispatcher	return values in UCmds / UUCmds
java.sql.Statement	IMovie / IUserMovieList / IUserDatabase

- The external ports of the subproblem architectures and the global architecture correspond to the interfaces and connection types in the technical context diagram.

external port type	**interface in architecture**	**required/provided**	**interface in technical context diagram**
UserPort	javax.servlet.http.HttpServlet	provided	AT!{doGet, doPost}
	javax.servlet.RequestDispatcher	required	MR!{forward}
UnregisteredUserPort	javax.servlet.http.HttpServlet	provided	AT!{doGet, doPost}
	javax.servlet.RequestDispatcher	required	MR!{forward}
MAdapterPort	java.sql.Statement	required	MR!{executeQuery, executeUpdate}
UMLAdapterPort	java.sql.Statement	required	MR!{executeQuery, executeUpdate}
UDAdapterPort	java.sql.Statement	required	MR!{executeQuery, executeUpdate}

2.2 D2

2.2.1 Inter-Component Interaction

- forwardRegister

context MRA_Register::forwardRegister(userData: UserData)
pre: true
post: if userData.age >= 18 and not ud – > one (u: userDatabase|userData.username = u.userData.username) then
ud->one(u:UserDatabase | u.UserData = userData) and ud->size() = ud @pre->size()+1 and
webpageUnregisteredUserˆshowOk()
else webpageUnregisteredUserˆshowFail()
end if

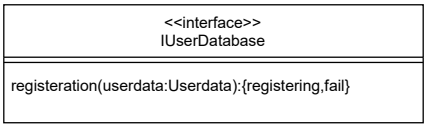

Figure 2.12: Operation specification

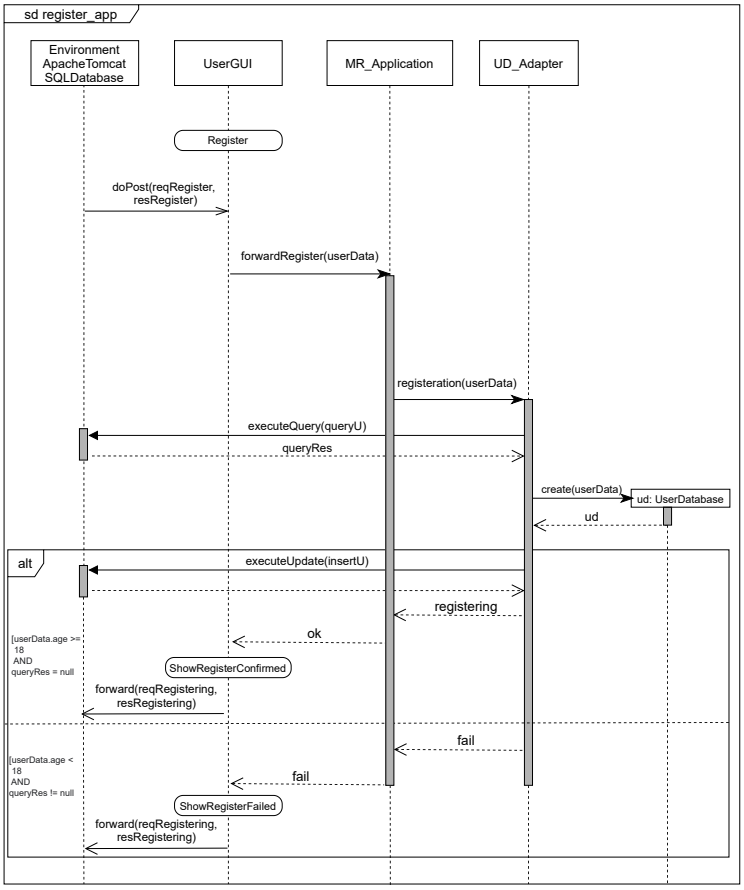

Figure 2.13: Interface Behavior

Remarks:

- reqRegister represent HTTPServletRequest objects containing the required user input.
- resRegister represent HttpServletResponse objects as the counterpart for the request.
- The state predicate Register represents that the the input form for registration is shown.
- The state predicate ShowRegisterConfirmed represents that the the confirmation is shown.
- The state predicate ShowRegisterFailed represents that an error message is shown.
- forward(...) sends the request and response back to the server to generate the HTML webpage.

queryU:

SELECT * from Userdatabase WHERE (userData.username=”userData.username”)

insertM:

INSERT INTO UserData (userData) VALUES (”userData”)

- addNewMovie

context MRA_AddNewMovie::createNewMovie(movieData: MovieData)
pre: true
post: if not mo ->one (m: Movie | movieData = m.movieData) then
m -> one(md:MovieData | md.movieData = movieData) and m -> size() = m@pre -> size()
+1 and
webpageUserˆshowConfirmAdd() else
webpageUserˆshowFailAdd()
end if

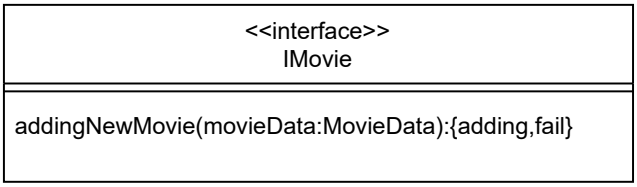

Figure 2.14: Operation specification

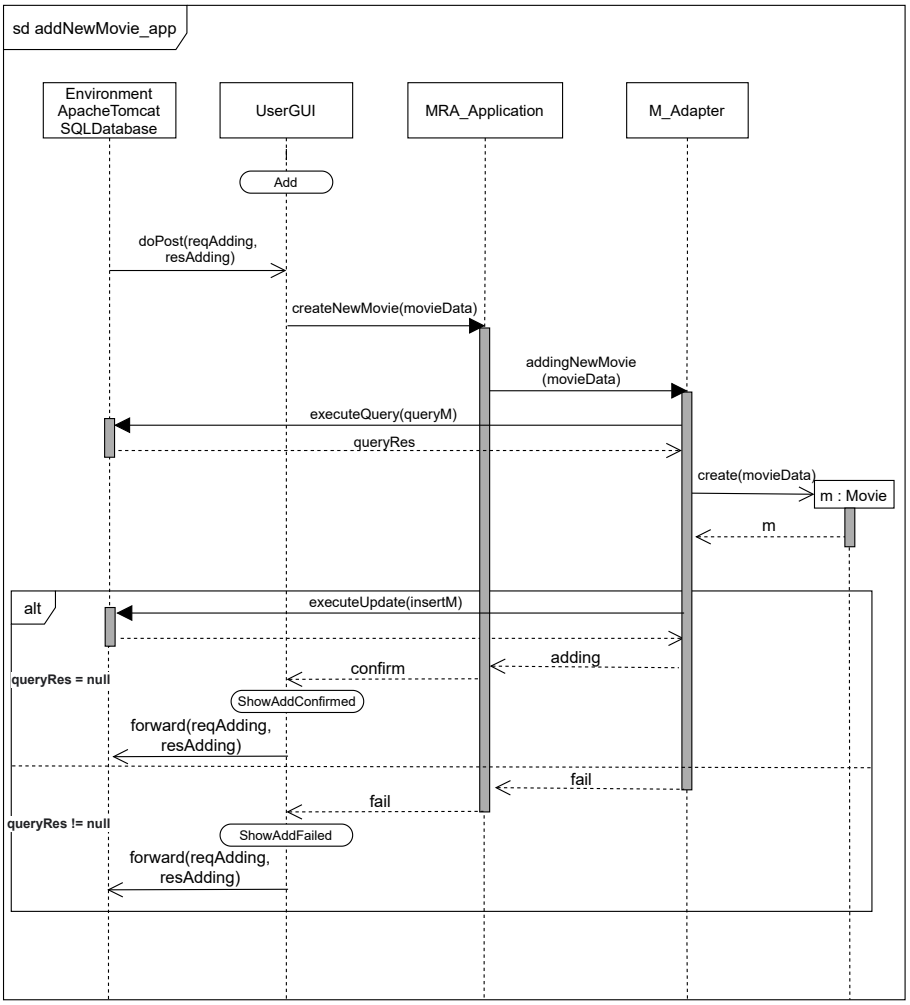

Figure 2.15: Interface Behavior

- reqAdding represent HTTPServletRequest objects containing the required user input.

- resAdding represent HttpServletResponse objects as the counterpart for the request.
- The state predicate Add represents that the the input form for adding the selected offer is shown.
- The state predicate ShowAddConfirmed represents that the the confirmation is shown.
- The state predicate ShowAddFailed represents that an error message is shown.
- forward(...) sends the request and response back to the server to generate the HTML webpage.

queryM

SELECT * from Movie WHERE (movieData="movieData")

insertM

INSERT INTO Movie (movieData) VALUES ("movieData")

- Rate

context MRA_Rate::forwardRating(movieRating::Float,mid::Integer,username::string)
pre: not uml -> exist (u: UserMovieList | mid = u.mid AND userData.username = u.username)
post: if 0 <= movieRating AND movieRating <= 10 then
uml -> any (u: UserMovieList | mid = u.mid AND
u.movieRating = movieRating AND
u.username = userData.username) AND
uml->size() = uml @pre->size() +1 AND
webpageUserˆShowRateOk()
else webpageUserˆShowFailOk()
end if

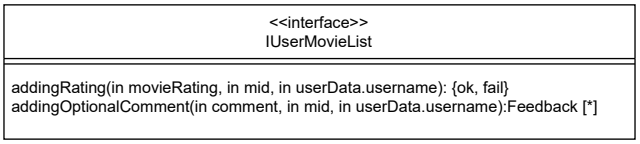

Figure 2.16: Operation specification

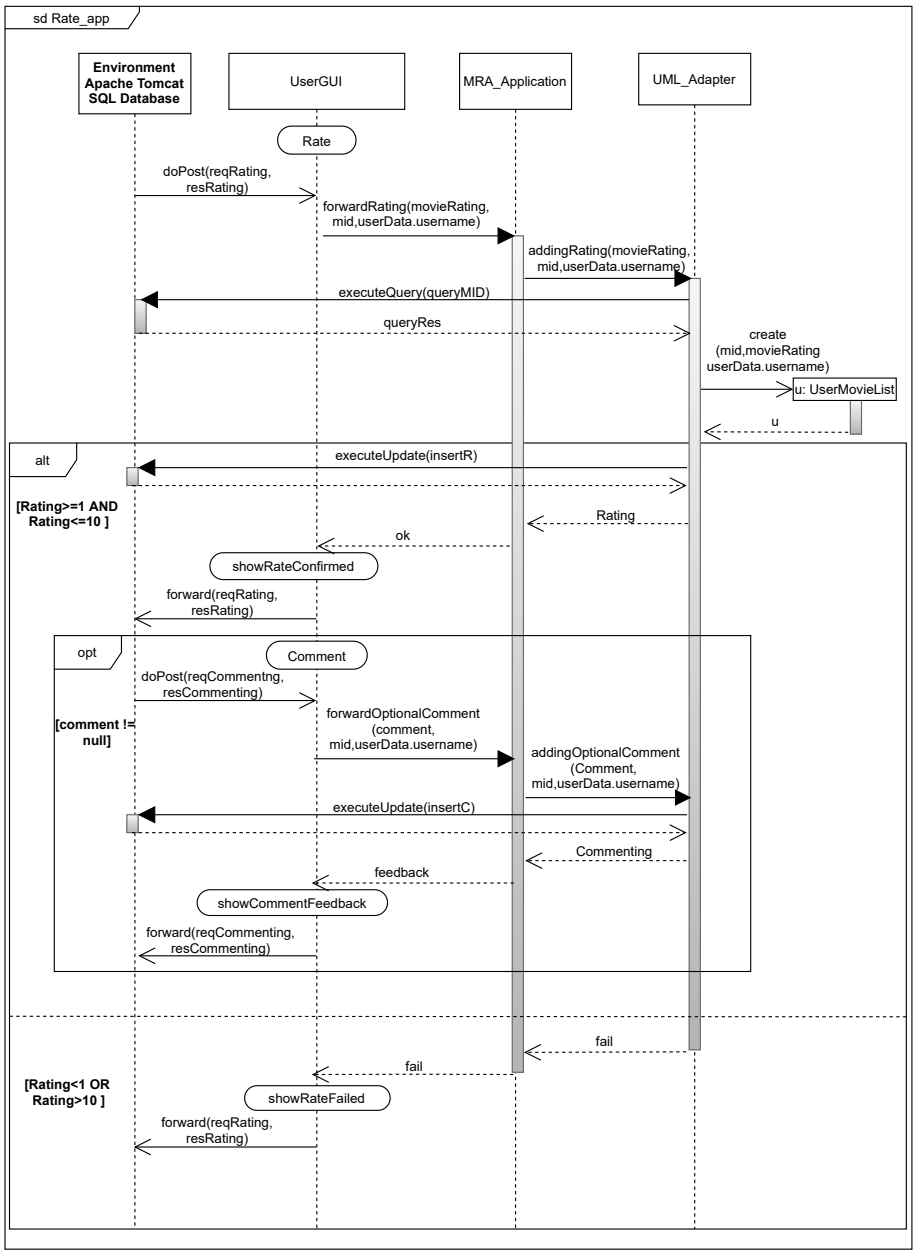

Figure 2.17: Interface Behavior

- reqSelect and reqRating represent HTTPServletRequest objects containing the required user input.
- resSelect and resRating represent HttpServletResponse objects as the counterpart for the request..
- The state predicate Rate represents that the the input form for Rating for the selected movie is shown.
- The state predicate ShowRateConfirmed represents that the the confirmation is shown..
- The state predicate ShowRateFailed represents that an error message is shown.
- forward(...) sends the request and response back to the server to generate the HTML webpage.

queryMID

SELECT * from Movie WHERE (mid = ”mid”))

insertR

INSERT INTO UserMovieList(movieRating, mid,username) VALUES(”u.movieRating”,”u.mid”,”u.username”)

insertC

UPDATE UserMovieList SET comment=u.comment WHERE mid=u.mid AND username = u.username

- ShowMovieOverview

context WebpageUser::getAllMovies()
pre: true
post:m:Movie.movieAvarageRating=uml:UserMovieListˆcalculateAvgRating (uml.movieRating,m.mid) let res: OrderedSet(Movie) = m->sortedBy(m:Movie | m.movieAvarageRating)
in webpageUserˆshowAllMovies(res)

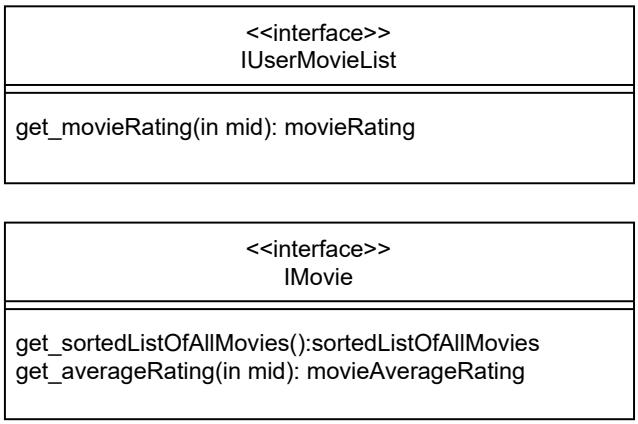

Figure 2.18: Operation specification

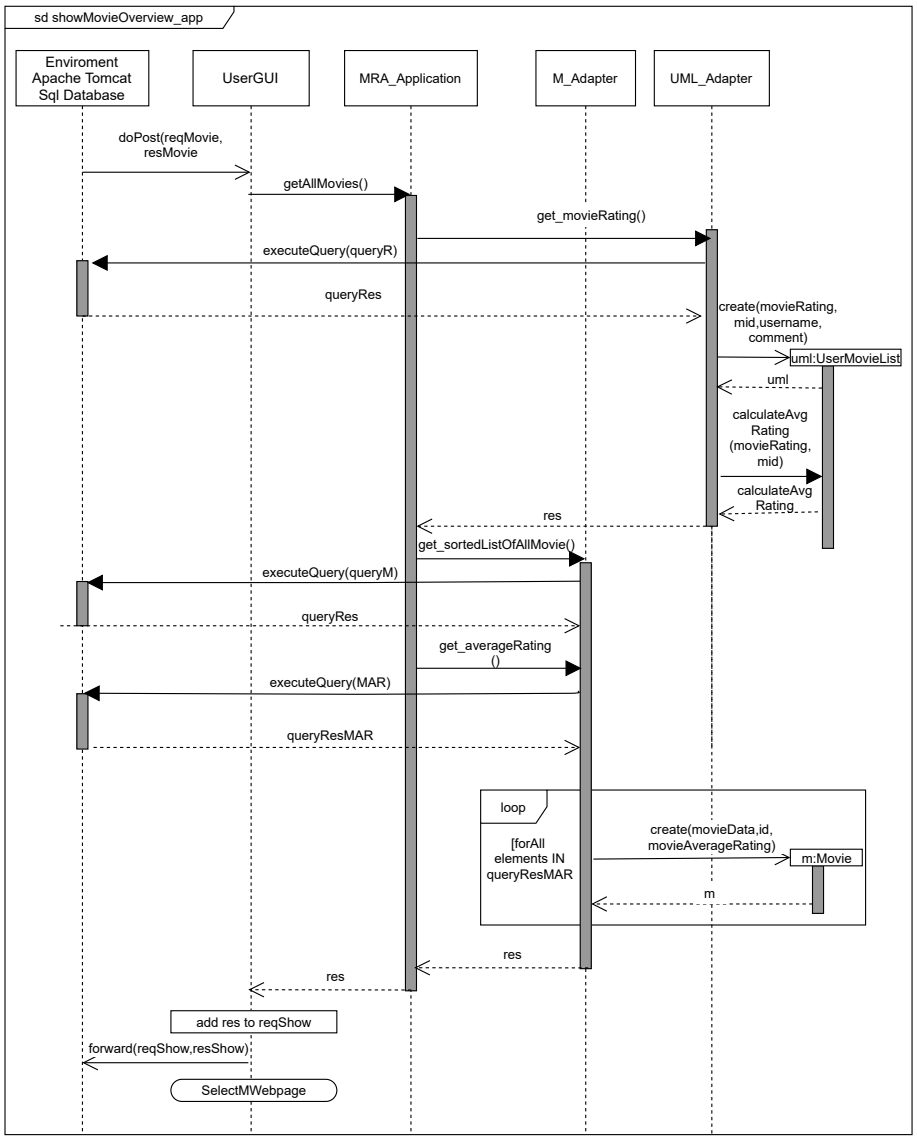

Figure 2.19: Interface Behavior

- reqSelect and reqRating represent HTTPServletRequest objects containing the required user input.
- resSelect and resRating represent HttpServletResponse objects as the counterpart for the request.
- m refers to an object of class Movie.
- forward(...) sends the request and response back to the server to generate the HTML webpage.

queryR

SELECT * from UserMovieList WHERE (movieRating=”movieRating”)

queryM

SELECT * from Movie GROUP BY m.mid

2.2.2 Validation

- Consistency of sdRegister_app and sdRegister.

Message in D2	Corresponding message in A3
doPost(reqRegister, resRegister)	refines register
forwardRegister(...)	forwardRegister
registeration(...)	registeration
executeQuery(queryU)	refines registeration
executeUpdate(updateU)	refines registeration
forward(reqRegistering,resRegistering)	okRepresentation
forward(reqRegistering,resRegistering)	failAgeRepresentation

- Consistency of sdAddNewMovie_app and sdAddNewMovie.

Message in D2	Corresponding message in A3
doPost(reqAdding, resAdding)	refines addNewMovie
createNewMovie(...)	createNewMovie
addingNewMovie(...)	addingNewMovie
executeQuery(queryM)	refines addingNewMovie
executeUpdate(updateM)	refines addingNewMovie
forward(reqAdding,resAdding)	confirmAddRepresentation
forward(reqAdding,resAdding)	failAddRepresentation

- Consistency of sdshowMovieOverview_app and sdshowMovieOverview.

Message in D2	Corresponding message in A3
doPost(reqMovie, resMovie)	refines viewAllMovies
getAllmovies(...)	getAllmovies
get_movieRating(...)	get_movieRating
get_sortedListOfAllMovies(...)	get_sortedListOfAllMovies
get_averageRating(...)	get_averageRating
executeQuery(queryR)	refines get_movieRating
executeQuery(queryM)	refines get_sortedListOfAllMovies
executeQuery(queryMAR)	refines get_averageRating
calculateAvgRating(...)	refines get_movieRating
forward(reqMovie,resMovie)	sortedMovieRepresentation

- Consistency of sdRate_app and sdRate.

Message in D2	Corresponding message in A3
doPost(reqRating, resRating)	refines addRating
forwardRating(...)	forwardRating
addingRating(...)	addingRating
executeQuery(queryMID)	refines addingRating
executeUpdate(InsertR)	refines addingRating
forward(reqRating,resRating)	RateOkRepresentation
forward(reqRating,resRating)	RateFailRepresentation
doPost(reqCommenting, resCommenting)	refines addOptionalComment
forwardOptionalComment(...)	forwardOptionalComment
addingOptionalComment(...)	addingOptionalComment
executeUpdate(InsertC)	refines addingOptionalComment
forward(reqCommenting,resCommenting)	feebackOptionalCommentRepresentation

- Consistency with life-cycle
  - Register+, Rate+, AddNewMovie+, ShowMovieOverview+ : sdregister_app, sdRate_app, sdrAddNewMovie_app and sdShowMovieOverview_app can be executed an arbitrary number of times without a precondition.
  - addRating+;[addOptionalComment]: After adding a rating in sdRate_app UserGUI is in the state Comment, such that the addRate and addOptionalComment can be executed in sequence as described by the life-cycle.

- forwardRegister(...) is realized in sdRegister_app
  - **Precondition** does not have to be established, because it is true.
  - **Postcondition** is established, because the selected username is first searched in the database by SQL command queryU and then it is checked if it is available. If it is available, then a corresponding Registration is added using insertU . The UserGUI is instructed to show the confirmation by the return value ok. If it is not available then the UserGUI is instructed to show the fail information by the return value fail

- forwardRating(...) is realized in sdRating_app
  - **Precondition** does not have to be established, because it is true.
  - **Postcondition** is established, because it is first searched in the database by SQL command queryMID and then it is checked if there is a rating from this user on this movie. if the result is null , then a corresponding Rating is added using insertR. The UserGUI is instructed to show the confirmation by the return value ok. If there is a rating already then the UserGUI is instructed to show the fail information by the return value fail.

- createNewMovie(...) is realized in sdaddNewMovie_app
  - **Precondition** does not have to be established, because it is true.
  - **Postcondition** is established, because the addedMovie is first searched in the database by SQL command queryM and then it is checked if it exists. If it is not, then the Movie is added using insertM and The UserGUI is instructed to show the confirmation by the return value ok. If it exists. then the UserGUI is instructed to show the fail information by the return value fail.

- ShowMovieOverview(...) is realized in sdshowMovieOverview_app
  - **Precondition** does not have to be established, because it is true.
  - **Postcondition** MR_Application delegates the message to M_Adapter and UML_Adapter. Using the SQL command queryR and queryM, M_Adapter and UML_Adapter. selects all Ratings and All movies Then UML_Adapter calculates the average rating using calculateAvgRating(..) for each movie and returns it to MR_Application that forwards the result to UserGUI. That realizes webpageUserˆshowAllMovies(res).

- All messages in the application interface classes of Step D1: Software architecture must be used in some sequence diagram.

Interface	**Message**	**Used in sequence Diagram**
UUCmds	forwardRegisteRegister	sd register_app
Ucmds	forwardRating	sd rate_app
	forwardOptionalComment	sd rate_app
	forwardOptionalComments	sd rate_app
	CreateNewMovie	sd AddNewMovie_app
	forwardRating	sd AddNewMovie_app
	forwardOptionalComment	sd AddNewMovie_app
	forwardRating	sd showMovieOverview_app
	forwardOptionalComments	sd showMovieOverview_app
	createNewMovie	sd showMovieOverview_app
	getAllMovies	sd showMovieOverview_app
IUserDatabase	get_ userInfo	sd register_app
	registration	sd register_app
IUserMovieList	get_userListOFMovies	sd Rate_app
	addingRating	sd Rate_app
	addingOptionalComment	sd Rate_app
	addingRating	sd showMovieOverview_app
	get_userListOFMovies	sd showMovieOverview_app
	addingOptionalComment	sd showMovieOverview_app
	get_movieRating	sd showMovieOverview_app
IMovie	get_sortedListOfAllMovies	sd addNewMovie_app
	addingNewMovie	sd addNewMovie_app
	get_sortedListOfAllMovies	sd showMovieOverview_app
	addingNewlMovies	sd addNewMovie_app
	get_averageRating	sd addNewMovie_app

- The directions of messages must be consistent with the required and provided interfaces of Step D1: Software architecture.

Interfac **Message**	**Provided by** **Recipient**	**Required by** **Sender**
UUCmds	MRA_Application	UnregisteredUserGUI
forwardRegister()	MRA_Application	UnregisteredUserGUI
IUserDatabase	UDAdapter	MRA_Application
registeration(userData)	UDAdapter	MRA_Application
UCmds	MRA_Application	UserGUI
forwardRating(movieRating,mid,userData.username)	MRA_Application	UserGUI
forwardOptionalComment(comment, mid,userData.username)	MRA_Application	UserGUI
getAllMovies()	MRA_Application	UserGUI
IUsermovielist	UML_Adapter	MRA_Application
addingRating(movieRating, mid,userData.username)	UML_Adapter	MRA_Application
addingOptionalComment(comment, mid,userData.username)	UML_Adapter	MRA_Application
get_userListOfMovies()	UML_Adapter	MRA_Application
IMovie	M_Adapter	MRA_Application
addingNewMovie(movieData)	M_Adapter	MRA_Application
get_sortedListOfAllMovies()	M_Adapter	MRA_Application
get_averageRating()	M_Adapter	MRA_Application

- Messages must connect components as connected in the software architecture of Step D1: Software architecture.

Component	**Connected components in architecture**	**Connected components in sequence Diagramm**
MRA_Application	UnregisteredUserGUI, UserGUI, M_Adapter,UML_Adapter, UD_Adapter	UnregisteredUserGUI, UserGUI,M_Adapter, UML_Adapter, UD_Adapter
UD_Adapter	MRA_Application, Enviroment	MRA_Application, Enviroment
UMl_Adapter	MRA_Application, Enviroment	MRA_Application, Enviroment
M_Adapter	MRA_Application, Enviroment	MRA_Application, Enviroment
UnregisteredUserGUI	MRA_Application, Enviroment	MRA_Application, Enviroment
UserGUI	MRA_ Application ,Enviroment	MRA_ Application ,Enviroment

2.3 D3

2.3.1 Intra-Component Interaction

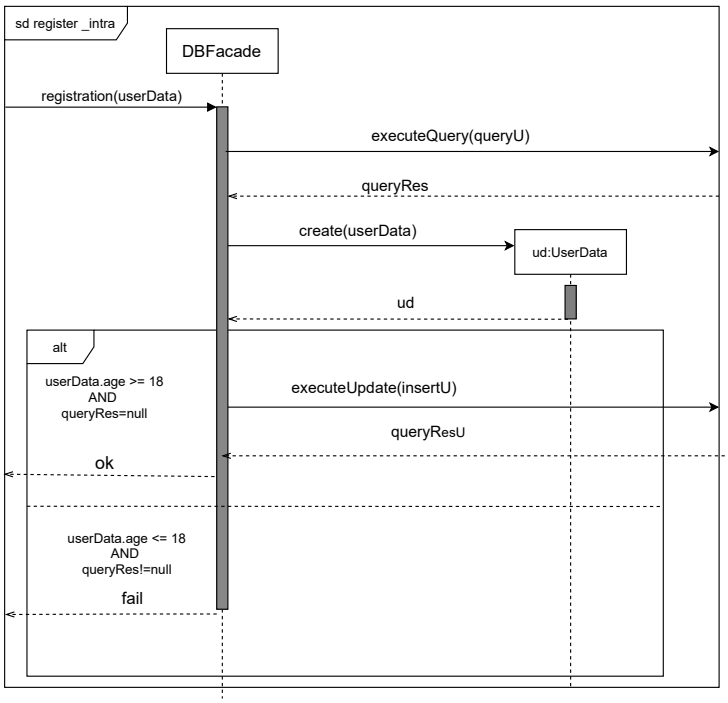

Figure 2.20: Register Intra-Component Interaction

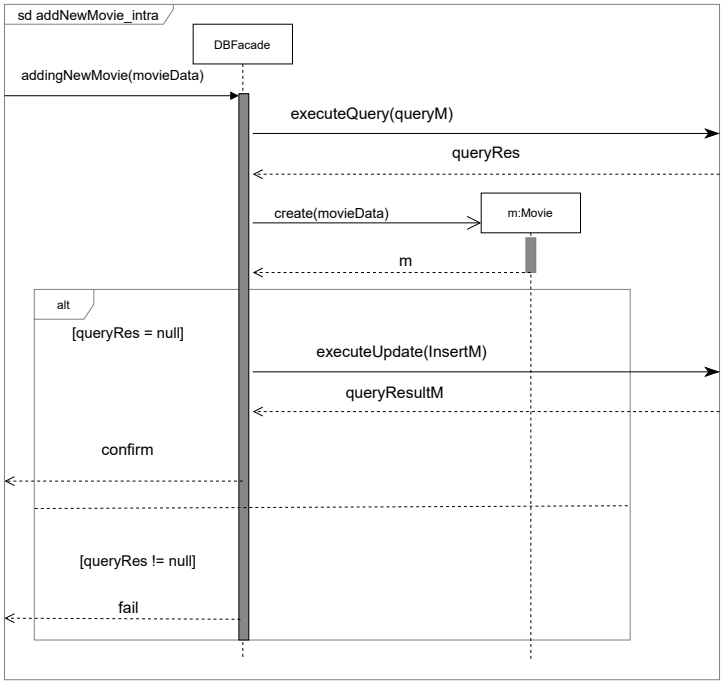

Figure 2.21: Adding New Movie Intra-Component Interactionr

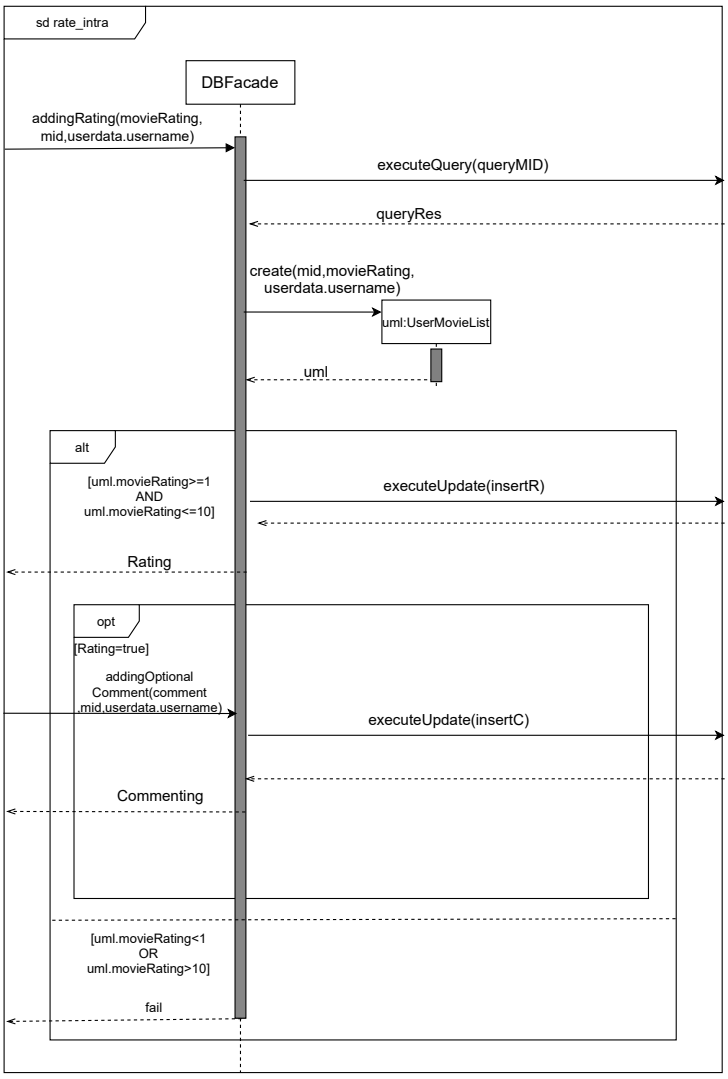

Figure 2.22: Rating Intra-Component Interaction

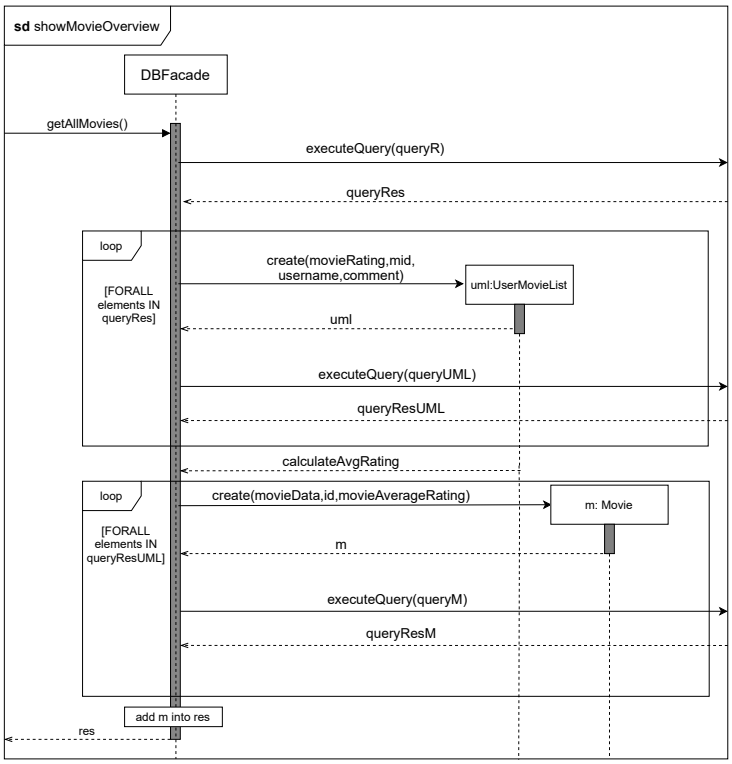

Figure 2.23: Show Movie Overview Intra-Component Interaction

2.3.2 Validation

- **Sequence diagrams of one component must be consistent with the corresponding interface behavior in step D2: Inter-Component Interaction.**

Messages of the SD register_intra in D3	Messages of the SD register_app in D2
registration(...)	registration(...)
executeQuery(queryU)	executeQuery(queryU)
create(...)	Refinment
executeUpdata(insertU)	Refinment

Messages of the SD addNewMovie_intra in D3	Messages of the SD addNewMovie_app in D2
addingNewMovie(...)	addingNewMovie(...)
executeQuery(queryM)	executeQuery(queryM)
create(...)	Refinment
executeUpdaten(InsertM)	executeUpdaten(InsertM)

Messages of the SD rate_intra in D3	Messages of the SD rate_app in D2
addingRating(...)	addingRating(...)
executeQuery(queryMID)	executeQuery(queryM)
create(...)	create(...)
executeUpdate(insertR)	executeUpdate(insertR)
addingOptionalComment(...)	addingOptionalComment(...)
executeUpdate(inserC)	executeUpdate(inserC)

Messages of the SD showMovieOverview_intra in D3	Messages of the SD showMovieOverview_app in D2
getAllMovies(...)	getAllMovies(...)
executeQuery(queryR)	executeQuery(queryR)
create(...)	Refinment
executeQuery(queryM)	executeQuery(queryM)
executeQuery(queryUML)	executeQuery(queryUML)

- **It must be possible to relate any new state (predicates) to the state predicates of Step D2: Inter-Component Interaction.**

New state (predicates) in step D3: Intra-Component Interaction	State predicates in step D2: Inter- Component Interaction
-	
-	

No new state (predicates) is introduced in step D3: Intra-Component Interaction.

2.4 D4

2.4.1 State Machine UserGUI

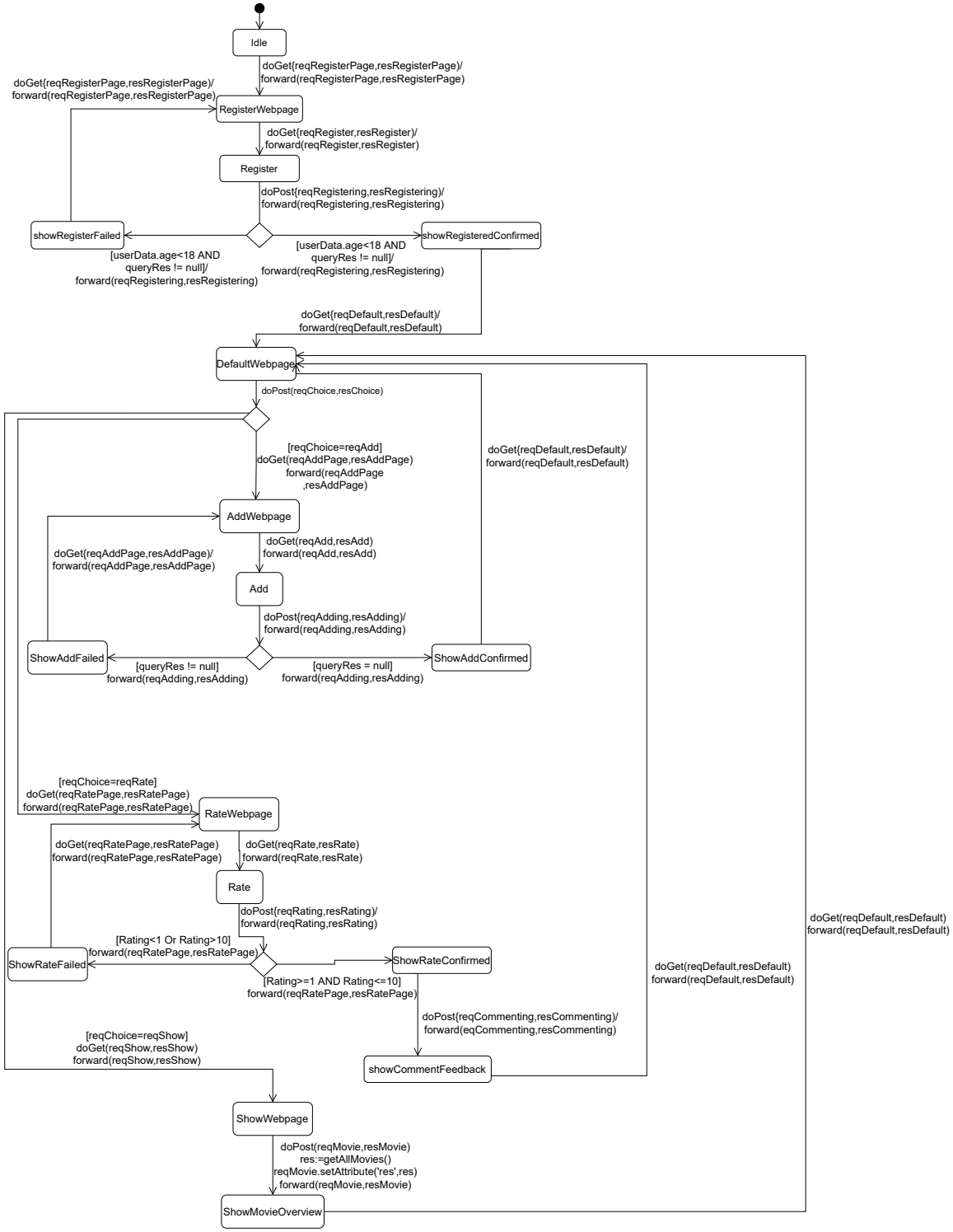

Figure 2.24: State Machine for UserGUI

2.4.2 Validation

- **The state machines describe the same behavior as in Step D2: Inter-Component Interaction or Step D3: Intra-Component Interaction**

Component UserGUI					
Source State	**Target State**	**Input Signal**	**Mapped to Message**	**Output Signal**	**Mapped to Message**
init	Register Webpage	doGet(RegisterPage, resRegisterPage)	-	forward(req RegisterPage, resRegisterPage)	-
Register Webpage	Register	doGet(reqRegister, resRegister)	-	forward(req Register, ,resRegister)	-
Register	showRegister Failed	doPost(req..)	doPost(req..)	forward(req..)	forward(req..)
Register	showRegister confirmed	doPost(req..)	doPost(req..)	forward(req..)	forward(req..)
showRegister failed	Register Webpage	doGet(req..)	-	forward(req..)	-
showRegister confirmed	Default Webpage	doGet(req..)	-	forward(req..)	-
Default Webpage	Add Wepage	doGet(req..)	-	forward(req..)	-
Default Webpage	Rate Wepage	doGet(req..)	-	forward(req..)	-
Default Webpage	Show Wepage	doGet(req..)	-	forward(req..)	-
Add Webpage	Add	doGet(req..)	-	forward(req..)	-
Add Webpage	ShowAdd failed	doPost(req Adding.)	doPost(req Adding.)	res=adding NewMovie(.)	adding NewMovie(.)
Add Webpage	ShowAdd failed	doPost(req Adding.)	doPost(req Adding.)	res=adding NewMovie(.) forward(reqAdd)	adding NewMovie(.) forward(reqAdd
Add Webpage	ShowAdd confirmed	doPost(req Adding.)	doPost(req Adding.)	res=adding NewMovie(.) forward(req, add)	adding NewMovie(.) forward(req, ,add)
ShowAdd confirmed	Default Webpage	doGet(req..)	-	forward(req..)	-
Rate Webpage	ShowRate failed	doPost(req Rating.)	doPost(req Rating.)	res=adding Rating(.) forward(req, rate)	adding Rating(.) forward(req rate)
Rate Webpage	ShowRate confirmed	doPost(req Rating.)	doPost(req Rating.)	res=adding Rating(.) forward(req, rate)	adding Rating(.) forward(req rate)
show webpage	show webpage	doPost(req Movie	doPost(req Movie	res:=get AllMovies() forward(reqMovie ,resMovie)	get_sorted ListOfAllMovie()

- **The state machines are consistent with the life-cycle model of Step A6: Software lifecycle.**

  All states are covered by a life-cycle

Component UnregisterdUserGUI	
$LC_{unregisteredUser} = (Register)^+$	
State	**Covered by Life Cycle Part**
init	Register
RegisterWebPage	Register
Register	Register
ShowRegisterFailed	Register
ShowRegisterConfirmed	Register

Component UserGUI	
$LC_{User} = (AddNewMovie \mid ShowMoviesOverview \mid AddRating; [AddOptionalComment])^*$	
State	**Covered by Life Cycle Part**
DefaultWebPage	AddNewMovie,ShowMovieOverview,AddRating
AddWebpage	AddNewMovie
Add	AddNewMovie
ShowAddFailed	AddNewMovie
ShowAddConfirmed	AddNewMovie
RateWebpage	AddRating
Rate	AddRating
ShowRateFailed	AddRating
ShowRateConfirmed	AddRating
ShowCommentFeedback	AddOptionalComment
ShowWebPage	ShowMovieOverview
ShowMovieOverview	ShowMovieOverview

- **All Transitions are covered by a life-cycle**

Component UnregisteredUserGUI				
$LC_{unregisteredUser} = (Register)^+$				
Source State State	**Target State**	**Input Signal**	**Output Signal**	**life cycle Part**
init	Register Webpage	doGet(RegisterPage, resRegisterPage)	forward(req RegisterPage, resRegisterPage)	Register
Register Webpage	Register	doGet(reqRegister, resRegister)	forward(req Register, ,resRegister)	Register
Register	showRegister Failed	doPost(req..)	forward(req..)	Register
Register	showRegister confirmed	doPost(req..)	forward(req..)	Register
showRegister failed	Register Webpage	doGet(req..)	forward(req..)	$(Register)^+$
showRegister confirmed	Default Webpage	doGet(req..)	forward(req..)	$(Register)^+$
Component UnregisteredUserGUI				
$LC_{User} = (AddNewMovie\|ShowMoviesOverview\|AddRating; [AddOptionalComment])^*$				
Source State State	**Target State**	**Input Signal**	**Output Signal**	**life cycle Part**
Default Webpage	Add Wepage	doGet(req..)	forward(req..)	AddNewMovie
Default Webpage	Rate Wepage	doGet(req..)	forward(req..)	AddRating
Default Webpage	Show Wepage	doGet(req..)	forward(req..)	ShowMovie Overview
Add Webpage	Add	doGet(req..)	forward(req..)	AddNewMovie
Add Webpage	ShowAdd failed	doPost(req Adding.)	res=adding NewMovie(.)	AddNewMovie
Add Webpage	ShowAdd failed	doPost(req Adding.)	res=adding NewMovie(.) forward(reqAdd)	AddNewMovie
Add Webpage	ShowAdd confirmed	doPost(req Adding.)	res=adding NewMovie(.) forward(reqAdd,.)	AddNewMovie
ShowAdd confirmed	Default Webpage	doGet(req..)	forward(req..)	AddNewMovie
ShowAdd failed	Default Webpage	doGet(req..)	forward(req..)	AddNewMovie
Rate Webpage	ShowRate failed	doPost(req Rating.)	res=adding Rating(.) forward(req,	AddRating
Rate Webpage	ShowRate confirmed	doPost(req Rating.)	res=adding Rating(.) forward(reqRate)	AddRating Rating(.) forward(reqRate)
ShowRate confirmed	RateWebPage Webpage	doGet(req..)	forward(req..)	AddRating
ShowRate	RateWebPage	doGet(req..)	forward(req..)	AddRating

failed	Webpage			
ShowComment feedback	DefaultWebPage Webpage	doGet(req..)	forward(req..)	AddRating
show webpage	show webpage	doPost(req Movie	res:=get AllMovies() forward(reqMovie ,resMovie)	get_sorted ListOfAllMovie()

3 Glossary

Table 3.1: Glossary

Name	Type	Description	Source
A			
actorsName	attribute	represents the name of actor	class model
Add	state predicate	Creating a new movie entry in the movies database	sd addNew-Movie_app, state machine User-GUI
Adding	message	Creating a new movie entry in the movies database	sd addNew-Movie_app
addingMovieToList	phenomena	Saving a new movie in the users personal movie list.	CD
addingNewMovie()	message	Add a new movie if it is not in the list of all movies	sd addNew-Movie_intra
addNewMovie	phenomena,auxiliary function	Add a new movie if it is not in the list of all movies.	CD, pdAddNew-Movie, sdAddNew-Movie, class model,sd addNew-Movie_app
addOptionalComment	phenomena, auxil-iary function	Add an optional comment along with rating to an existing movie.	CD, pdRate, sdRate, class model,sd Rate_app
addingRating	phenomena, auxil-iary function	Adding new movie rating to the movies database.	CD, pdRate, sdRate, sub-ArchRate,sd Rate_app,sd rate_intra
addingUsersToGroup	phenomena	Marking added users in the group database.	CD
addOptionalComment	phenomena	Add an optional comment along with rating to an existing movie.	CD, pdRate, sdRate, class model
addMovieToList	phenomena	Add an a movie to users personal movie list.	CD

Table 3.1: Glossar

Name	Type	Description	Source
addNewMovie	phenomena	Add a new movie if it is not in the list of all movies.	CD, pdAddNew-Movie, sdAddNew-Movie, class model
addRating	phenomena	Add a rating to an existing movie.	CD, pdRate, sdRate, class model
addUsersToGroup	phenomena	Add a registered user to a discussion group.	CD
AddWebpage	state	its connection between the user and the machine	state machine User-GUI
age	attribute	represents the age of the user	class model
AgeOutOfRange	state predicate	the users age is less than eighteen.	sdRegister
averageRatingShown	phenomena	the average rating of each movie displayed on the user's webpage along with the movies	pdShowMovie-Overview, sdShowMovie-Overview
ApacheTomcat	connection domain	An Open Source JSP and Servlet Container from the Apache Foundation.	TCD
api		Abbreviation for applecation program Interface	TCD
B			
banMember	phenomena	Ban a group member from a group if they misbehave.	CD
banningMember	phenomena	Removing a banned member from the group database.	CD
C			
calculateAvgRating	message	Calculate the Average	sd show-Movie Overview _intra
Comment	attribute,state predicate	A logged in user can optionally add a comment	class model,sd Rate_app
Commenting	message	A logged in user can optionally add a comment	sd rate_intra
confirm	message	show the confirmation	sd addNew-Movie_app, sd addNew-Movie_intra
confirmAddRepresentation	phenomena	The confirmation returned to the user.	pdAddNew-Movie, sdAddNew-Movie

Table 3.1: Glossar

Name	Type	Description	Source
create(userData)	auxiliary function	the user creates a new list in Data base	sd register_app,sd addNew-Movie_app ,sd Rate_app, sdshowMovie-Overview _app,sd register_intra,sd addNew-Movie_intra,sd rate_intra,sd showMovie Overview _intra
createNewMovie	phenomena,auxiliary function	Forwarding the user's request from webpage to movie database.	pdAddNew-Movie, sdAddNew-Movie, class model, subAr-chAddNew-Movie, sdaddNew-Movie_app
currentDate()	auxiliary function	Provides the current Date	Class model
D			
day	attribute	Represent the day	Class model
DB		Abbreviation for Database.	pfUpdate, pfQuery
Defaultwebpage	state	Indicates the starting page	state Ma-chine User-GUI
deletingGroup	phenomena	Removing a group from database if the administrator leaves.	CD
doGet	technical phe-nomenon	Called by the server to allow a servlet to handle a GET request.	TCD, subArch-ShowMovie-Overview
doPost	technical phe-nomenon,message	Called by the server to allow a servlet to handle a POST request	TCD, subArch-ShowMovie-Overview,sd register_app, sd addNew-Movie_app
director	attribute	represents the director of the movie	Class model
E			

Table 3.1: Glossar

Name	Type	Description	Source
email	attribute	represents the email address of the user	class model
EO		Abbreviation for Enquiry Operator.	pfQuery
executeQuery	technical phenomenon,message	Executes the given SQL statement, which returns a single ResultSet object.	TCD, subArch-ShowMovie-Overview,sd regis-ter_app,sd addNew-Movie_app,sd Rate_app,sd showMovie-Overview_app
executeQuery()	message	Executes the given SQL statement, which returns a single ResultSet object.	sd regis-ter_intra,sd addNew-Movie_intra,sd rate_intra,sd showMovie Overview _intra
executeUpdate	technical phenomenon,message	Executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement	TCD, subArch-ShowMovie-Overview,sd regis-ter_app,sd addNew-Movie_app,sd Rate_app,sd showMovie-Overview_app
executeUpdate()	message	Executes the given SQL statement, which may be an INSERT, UPDATE, or DELETE statement	sd regis-ter_intra,sd addNew-Movie_intra,sd rate_intra,sd showMovie Overview _intra
F			
fail	message	show fail message	sd regis-ter_intra,sd addNew-Movie_intra,sd rate_intra

Table 3.1: Glossar

Name	Type	Description	Source
failAddRepresentation	phenomena	The feedback returned to the user.	pdAddNew-Movie, sdAddNew-Movie
failAgeRepresentation	phenomena	An error displayed by Webpage-UnregistedUser upon registering with an invalid age.	pdRegister, sdRegister
feedbackGA	phenomena	Capture all the feedback going to the group administrator.	CD
feedbackGM	phenomena	Capture all the feedback going to the group member..	CD
feedbackOptional-CommentRepresentation	phenomena	responding back to the user for his comment request	pdRate, sdRate
feedbackU	phenomena	Capture all the feedback going to the user the user.	CD
feedbackUU	phenomena	Capture all the feedback going to the unregistered user.	CD
forward	technical phenomenon	Defines an object that receives requests from the client and sends them	TCD,sd register_app,sd addNew-Movie_app,sd Rate_app,sd showMovie-Overview_app
forwardOptionalComment	phenomena	Forwarding the user's comment from the connection domain to the machine	pdRate, sdRate, class model, sub-ArchRate, Rate_app
forwardRating	phenomena	Forwarding the user's rating from the connection to the machine	pdRate, sdRate, class model, sub-ArchRate,sd Rate_app
forwardRegister	phenomena	Forwarding the registration request from connection domain to the machine.	pdRegister, sdRegister, class model, subArchRegister,sd register_app
G			
G		Abbreviation for Group	CD
GA		Abbreviation for Administrator	CD

Table 3.1: Glossar

Name	Type	Description	Source
getAllMovies	phenomena, auxiliary function	forwarding user's request to see all movies from the connection domain to the machine.	pdShowMovieOverview, sdShowMovieOverview, subArchShowMovieOverview, sdshowMovieOverview _app,sd showMovie Overview _intra
get_averageRating	message, auxiliary function	the the machine receives average rating of each movie	sdShowMovieOverview,sd showMovieOverview_app
get_movieRating	message,auxiliary function	Machine gets users rating on a movie	sdShowMovieOverview, sdshowMovieOverview_app
get_sortedListOffAllMovies	message,auxiliary function	machine gets stored list off all Movies from Database	sdAddNewMovie, subArchAddNewMovie, sdshowMovieOverview_app
get_userInfo	message	user info given to the machine.	sdRegister, subArchRegister
get_userListOfMovies	message	user Movie List given to the machine.	sdRate, subArchRate
GM		Abbreviation for GroupMember	CD
Group	lexical-designed domain	Virtual movie discussion round representation.	CD
GroupAdministator	biddable domain	A user, who created and responsible for a group.	CD
groupInfo	phenomena	All info related to a group including group name and a list of members.	CD
GroupMember	biddable domain	User inside a group.	CD
gui	technical phenomenon	User interfaces of MailClient and HTML webpages	TCD
H			
http	technical phenomenon	defined in RFC 2616 (Network Working Group, 1999)	TCD
I			
id	attribute	represents unique id of Movie	Class model

Table 3.1: Glossar

Name	Type	Description	Source
Idle	state	Indicates thate the server waits for incoming requests	state Machine User-GUI
IMovie	interface	Interface for containing users movie	subArch-Add-NewMovie, subArch-ShowMovie-Overview, globalArch
IOD		Abbreviation for Input output Device.	pfUpdate, pfQuery
IUserDatabase	interface	Interface for database containing all the information related to users.	subArch-Register, globalArch
IUsermovielist	interface	Interface for database containing users movie lists	subArch-Rate,subArch-ShowMovie-Overview, globalArch
L			
LCMovieRating	life-cycle	Combined life-cycle (all users and the internal operation)	LC
LCunregisteredUser	life-cycle	life-cycle for one unregistered User	LC
LCUser	life-cycle	life-cycle for one User	LC
leaveOwnGroup	Phnomena	Group administrator leaving a group, which he created.	CD
leavingGroup	phenomena	Group member leaving a group, in which he was added in.	CD
loggedIn	phenomena	Recording a login in the database.	CD
loggedOut	phenomena	Recording a logout in the database.	CD
logIn	phenomena	The act of logging in to the website.	CD
logOut	phenomena	The act of logging out from the website.	CD
M			
M		Abbreviation for Movie	CD, pdAddNew-Movie
m	message	Abbreviation for Movie	sd addNew-Movie_intra,sd showMovie Overview _intra
MRA		Abbreviation for MovieRatingApp	CD

Table 3.1: Glossar

Name	Type	Description	Source
M_Adapter	component	Responsible to add new Movie	subArchAdd-NewMovie, subArchShow-Movie-Overview ,glob-alArch,sd addNew-Movie_app, sdshowMovie-Overview_app
MRA_AddNewMovie	domain	The machine domain represent-ing the sub-problem of adding a new movie to the list.	pdAddNew-Movie
MRA_Application	component		subArch-Register, sub-ArchRate, subArchAdd-NewMovie, subArch-ShowMovie-Overview, glob-alArch,sd regis-ter_app,sd addNew-Movie_app,sd Rate_app, sdshowMovie-Overview_app
MRA_Rate	machine	The machine that is responsible for rating a movie	pdRate
MRA_Register	machine	Part of the machine which han-dles the registration	pdRegister
MRA_ShowMovieOverview	domain	a machine domain representing the sub-problem of showing an overview of all movies	pdShowMovie-Overview
MRAANM		Abbreviation for MRA_AddNewMovie	pdRate
MRAR		Abbreviation for MRA_Register	pdRegister, pdRate
month	attribute	Represent the month	Class model
Movie	lexical-designed do-main	A database that contains all movies.	CD, pdShowMovie-Overview
MovieAdapter	component	Responsible to show movie overview	subArch-ShowMovie-Overview, globalArch

Table 3.1: Glossar

Name	Type	Description	Source
movieAverageRating	phenomena	An average number that is calculated from all the ratings submitted to a movie.	CD, pdShowMovie-Overview, sdShowMovie-Overview, subArchShow-Movie-Overview
movieComments	phenomena	Comments given by the users on a movie.	CD
MovieData	class	Represents the Data of a Movie	Class model
MovieInDB	state predicate	A movie stored in Database	sdAddNew-Movie
movieInfo	phenomena	All info related to a movie.	CD, pdAddNew-Movie
movieRating	phenomena	Returns users rating on a movie	CD, pdShowMovie-Overview, sdShowMovie-Overview, subArch-ShowMovie-Overview
O			
OK	message	show agreement message	sd register_intra
okRepresentation	phenomena	A feedback shown to the user by the WebpageUnregisteredUser upon successful registration.	pdRegister, sdRegister
OptionalCommentOnMovie	phenomena	Returns the users comment on movie.	CD
origPubDate	attribute	represents the orgin publishing date for a movie	Class model
Q			
QM		Abbreviation for Query Machine.	pfQuery
queryRes	message	show feedback message	sd addNew-Movie_app,sd Rate_app, sdshowMovie-Overview_app, sd register _intra,sd addNew-Movie_intra,sd rate_intra,sd showMovie Overview _intra

Table 3.1: Glossar

Name	Type	Description	Source
queryResM	message	show feedback message from movie list	sd addNew-Movie_intra,sd showMovie Overview _intra
queryResU	message	show feedback message from User	sd regis-ter_intra
queryResUML	message	show feedback message from User movie list	sd show-Movie Overview _intra
R			
Rate	state predicate	Rating of the movie	sd Rate_app, state ma-chine User-GUI
rateFailRepresentation	phenomena	feedback shown to the user by the WebpageUser upon faild rat-ing.	pdRate, sdRate
RateWebpage	state	its a connection between the user and machine	state ma-chine User-GUI
Rating	message	Recording the movie rating in the database	sd rate_intra
rateOkRepresentation	phenomena	feedback shown to the user by the WebpageUser upon unsuc-cessful rating.	pdRate, sdRate
RatingAdded	state predicate	A movie get rating	sdRate
ratings	attribute	Recording the movie rating in the database	Class model
RatingInRange	state predicate	user can rat the movie	sdRate
RatingOutOfRange	state predicate	A movie has already rated	sdRate
RegSuccess	state predicate	the users info is added to the UserDatabase.	sdRegister
register	phenomena, state predicate,state	Submit information about user in order to use app functionality.	CD, pdReg-iste, sdReg-ister, class model,sd regis-ter_app,state machine UserGUI
registration	phenomena, auxil-iary function	Marking a successful registration in users database.	CD, pdReg-iste, sdReg-ister, sub-ArchRegis-ter ,sd reg-ister_app,sd register_intra

Table 3.1: Glossar

Name	Type	Description	Source
RegisterWebpage	state	it is a connection between the biddable domain and the machine	state machine UserGUI
registrationData	class	describing the registration information	Class model
RU		Abbreviation for RegisteredUser	CD
S			
sendChatMessage	phenomena	A group member sending a message in group chat.	CD
sendingChatMessage	phenomena	A group member message being forwarded to the group database.	CD
ShowAddConfirmed	state predicate	show Add confirm message	sd addNewMovie_app, state machine UserGUI
ShowAddFailed	state predicate	show Add failed message	sd addNewMovie_app, state machine UserGUI
showCommentFeedback	state predicate	show Comment Feedback message	sd Rate_app, state machine UserGUI
showRateConfirmed	state predicate	show Rate confirm message	sd Rate_app, state machine UserGUI
showFail()	auxiliary function	A response by the machine upon a faild registration	Class model
showFail	phenomena	A response by the machine upon a failed registration.	pdRegister, sdRegister
showFailAdd	phenomena	Displaying the failure corresponding to user's request.	pdAddNewMovie, sdAddNewMovie
showOk()	auxiliary function	A response by the machine upon a successful registration	Class model
showOk	phenomena	A response by the machine upon a successful registration.	pdRegister, sdRegister
showRateFailed	state predicate	show Rate failed message	sd Rate_app, state machine UserGUI
ShowRegisterConfirmed	state predicate,	show registeration confirm message	sd register_app,state machine UserGUI

Table 3.1: Glossar

Name	Type	Description	Source
ShowRegisterFailed	state predicate	show registeration failed message	sd register_app,state machine UserGUI
SelectMWebpage	state predicate	select movie webpage	sdshowMovie-Overview_app
sortedListOfAllMovies	phenomena	All movies in database sorted according to rating in descending order.	CD, pdShowMovie-Overview, pdAddNew-Movie, sdAddNew-Movie, sdShowMovie-Overview
sortedMessage	phenomena	All messages in group chat sorted according to sending time.	CD
sortedMovieRepresentation	phenomena	all the moving displayed on the user webpage sorted based on average rating.	pdShowMovie-Overview, sdShowMovie-Overview
showAllMovies	phenomena	Displaying all movies corresponding to the user's request.	pdShowMovie-Overview, sdShowMovie-Overview
showAverageRating	phenomena	Displaying average rating along with movies.	pdShowMovie-Overview, sdShowMovie-Overview
showConfirmAdd	phenomena	Displaying the confirmation corresponding to user's request.	pdAddNew-Movie, sdAddNew-Movie
showFeedbackOptional-Comment	phenomena	confirming adding additional comment process.	pdRate, sdRate
showRateFail()	auxiliary function	Displaying a confirmation that the rating is faild	Class model
showRateFail	phenomena	Displaying an error that the rating is out of range.	pdRate, sdRate
showRateOk()	auxiliary function	Displaying a confirmation that the rating is accepted	Class model
showRateOk	phenomena	Displaying a confirmation that the rating is accepted.	pdRate, sdRate
ShowWebpage	state	its a conniction between user and machine	, state machine UserGUI
signature	attribute	Represents user signature	Class model
SQLDatabase	causal Domain	Database tool that is tailored to suit specific needs of SQL developers.	TCD
T			

Table 3.1: Glossar

Name	Type	Description	Source
TimeData	class	Represents time(year and month and day)	Class model
title	attribute	Represents the title of the movie	class model
U			
U		Abbreviation for User	CD, pdRate, pdAddNew-Movie
UCmds	interface	User commands	subArch-Rate,sub-ArchAdd-NewMovie, globalArch, subArchShow-Movie-Overview
UD		Abbreviation for UserDatabase	CD , pdReg-ister, pdReg-ister,sd regis-ter_intra
UDAdapter	component	Responsible to create user ac-count	subArch-Register, glob-alArch,sd register_app
UDPort	component		subArch-Register, globalArch
UGUI	component	Web interface for User	subArchRate, globalArch
UM		Abbreviation for Update Ma-chine	pfUpdate
UML		Abbreviation for UserMovieList	CD, pdRate, pdShowMovie-Overview,sd rate_intra,sd showMovie Overview _intra
UML_Adapter	component	responsible to create user movie list	subArchRate, subArchShow-Movie-Overview, subArchShow-Movie-Overview, glob-alArch,sd Rate_app, sdshowMovie-Overview_app

Table 3.1: Glossar

Name	Type	Description	Source
UnregisteredUser	biddable domain	Person who can register to use apps functionality.	CD, pdRegister
UnregisteredUserGUI	component	Web interface for Unregistered User	subArchRegister, globalArch
unregisteredWebBrowser	connectionDomain	Web browser used by unregistered user	TCD
UO		Abbreviation for Update Operator	pfUpdate
User	biddable domain	A user who has already registered and can use the application functionality.	CD, pdShowMovieOverview
UserGUI	component	Web interface for User	subArchAddNewMovie, subArchShowMovieOverview, globalArch,sd register_app,sd addNewMovie_app,sd Rate_app, sdshowMovieOverview_app
userWebBrowser	connection domain	Web browser used by user, e.g. Opera.	TCD
UserData	class	Returns the informations related to user	Class model
UserDatabase	lexical-designed domain	A database containing all the information related to users.	CD, pdRegister
userInfo	phenomena	Returns all info related to user.	CD, pdRegister, sdRegister
userListOfMovies	phenomena	a function returning the list movies for a user	CD,pdRate, sdRate
UserMovieList	lexical-designed domain	A database containing users movie lists	CD, pdRate, pdShowMovieOverview
username	attribute	represents unque user name of user Data	class model
UsernameInDB	state predicate	A database containing the users name.	sdRegister
UPort	component		subArchRate, subArchShowMovie Overview, globalArch

Table 3.1: Glossar

Name	Type	Description	Source
UU		Abbreviation for UnregisteredUser	CD, pdRegister
UUCmds	interface	Unregistered User commands	subArch Register, globalArch
V			
viewAllMovies	phenomena	Look through all movies in the movie database.	CD, pdShowMovieOverview, sdShowMovieOverview
viewOthersMovieList	phenomena	View other group members movie list.	CD
W			
WebpageUnregisteredUser	domain	The connection domain between the unregistered user and the machine. Forwarding the inputs of the unregistered user to the machine and the outputs of the machine to the unregistered user.	pdRegister
WebpageUser	domain	The connection domain between the User and the machine.Forwarding the inputs of the user the machine and the outputs of the machine to the guest.	pdShowMovieOverview, pdAddNewMovie, pdRate
WU		Abbreviation for WebpageUser	pdRate, pdAddNewMovie
WUU		Abbreviation for WebpageUnregisteredUser.	pdRegister
Y			
year	attribute	Represent the year	Class model