UNIVERSITÄT DUISBURG ESSEN

Open-Minded

Lab for Software Engineering
Winter term 2020/2021

Lab for Software Engineering

Collaborative Calendar App

Ishrat Jahan (3001900)
Cedric Tala Temi (3101095)
Clara Tabea Voß (3079931)
Sofie Teresa Kalthof (3089808)
Joshua Redmann (3065754)
Björn Bravin (3065067)

March 6, 2021

University of Duisburg-Essen · Faculty of Engineering Working group Software Engineering – Prof. Dr. M. Heisel
Date: Di. 10-12 Supervisor: Roman Wirtz

Contents

1 Analysis **iii**
- 1.1 A1 . . . iii
  - 1.1.1 Requirements & Domain-Knowledge . . . iii
  - 1.1.2 Contextdiagram . . . v
  - 1.1.3 Validation . . . vi
- 1.2 A2 . . . x
  - 1.2.1 Problem Diagrams . . . x
  - 1.2.2 Problem frames . . . xv
  - 1.2.3 Validation . . . xvi
- 1.3 A3 . . . xxvii
  - 1.3.1 R10 . . . xxvii
  - 1.3.2 R11 . . . xxvii
  - 1.3.3 R14 . . . xxviii
  - 1.3.4 R12, R13, R15 . . . xxix
  - 1.3.5 Validation . . . xxix
- 1.4 A4 . . . xxxii
  - 1.4.1 Technical Context Diagram . . . xxxii
  - 1.4.2 Validation . . . xxxv
- 1.5 A5 . . . xxxvi
  - 1.5.1 AddAppointment . . . xxxvi
  - 1.5.2 AnswerAR . . . xxxviii
  - 1.5.3 ShowCalendar . . . xlii
  - 1.5.4 FinalizeAppointment . . . xliv
- 1.6 A6 . . . xlvi
  - 1.6.1 Validation I . . . xlvi
  - 1.6.2 Validation II . . . xlvi
  - 1.6.3 Validation III . . . xlvi
  - 1.6.4 Validation IV . . . xlvi
  - 1.6.5 Validation V . . . xlvii

2 Design **xlviii**
- 2.1 D1 . . . xlviii
  - 2.1.1 Validation . . . lvi
- 2.2 D2 . . . lvii
  - 2.2.1 AddAppointment . . . lvii
  - 2.2.2 AnswerAR . . . lix
  - 2.2.3 ShowCalendar . . . lxi
  - 2.2.4 FinalizeAppointment . . . lxii
  - 2.2.5 Validation . . . lxv
- 2.3 D3 . . . lxviii
- 2.4 D4 . . . lxix
  - 2.4.1 Validation . . . lxx

3 Glossary **lxxiii**

1 Analysis

1.1 A1

1.1.1 Requirements & Domain-Knowledge

Requirements

R1 Users should be able to register

R2 Registered users can login to gain access to their account

R3 If desired, users can log out of their account

R4 Registered users can create groups and are appointed as group administrator

R5 Application is available from different devices (Mobile, Desktop)

R6 Group administrators can invite other registered users to become a member of the group

R7 Group administrators can change a member's role

R8 Group administrators can remove members from the group

R9 Registered users can join the groups they are invited to

R10 Group members can create a new appointment request in the group's calendar and all dates on which the appointment can take place are added to the group calendar and marked as preliminary appointment dates

R11 A group member votes if a date is possible for them. They can suggest new dates for an appointment, even if one of the dates is possible for them

R12 If a date was found that is possible for all planned participants of an appointment request, then this date is selected to be the fixed date of the appointment and added as such to the calendar

R13 If the configured deadline date of an appointment request is reached and no date was found that is possible for all planned participants, then automatically a date is chosen which is possible for most of the planned participants. If there are several dates that are equally possible for all planned participants, the first possible date is selected

R14 All group members can access the group calendar containing all appointments of the group

R15 If a fixed date has been found, all preliminary dates are removed from the calendar and all planned participants for whom the date is possible are recorded as actual participants

Facts

F1 Group administrators are special group members

F2 Group members are special registered users

F3 Every appointment request has a name, description, location, duration, list of planned participants, a list of dates at which the appointment may take place, and a deadline after which the calendar app will automatically select an appropiate date for the appointment

Assumptions

A1 Every user can access the application from the internet

A2 Users have to be registered and logged in to use the functionalities provided by the application

A3 One calendar per group

A4 Registered users regularly check for group invites

A5 Group members regularly check for appointment requests

A6 Not all group members have to be planned participants

A7 Every created appointment request has no errors and fulfils all requirements

A8 Planned participants regularly check for newly suggested dates

A9 An appointment is found possible even if not all of the planned participants can participate

A10 If no date is possible, a planned participant has to suggest at least one new date for the appointment

A11 Planned participants have to select at least one of the existing dates

1.1.2 Contextdiagram

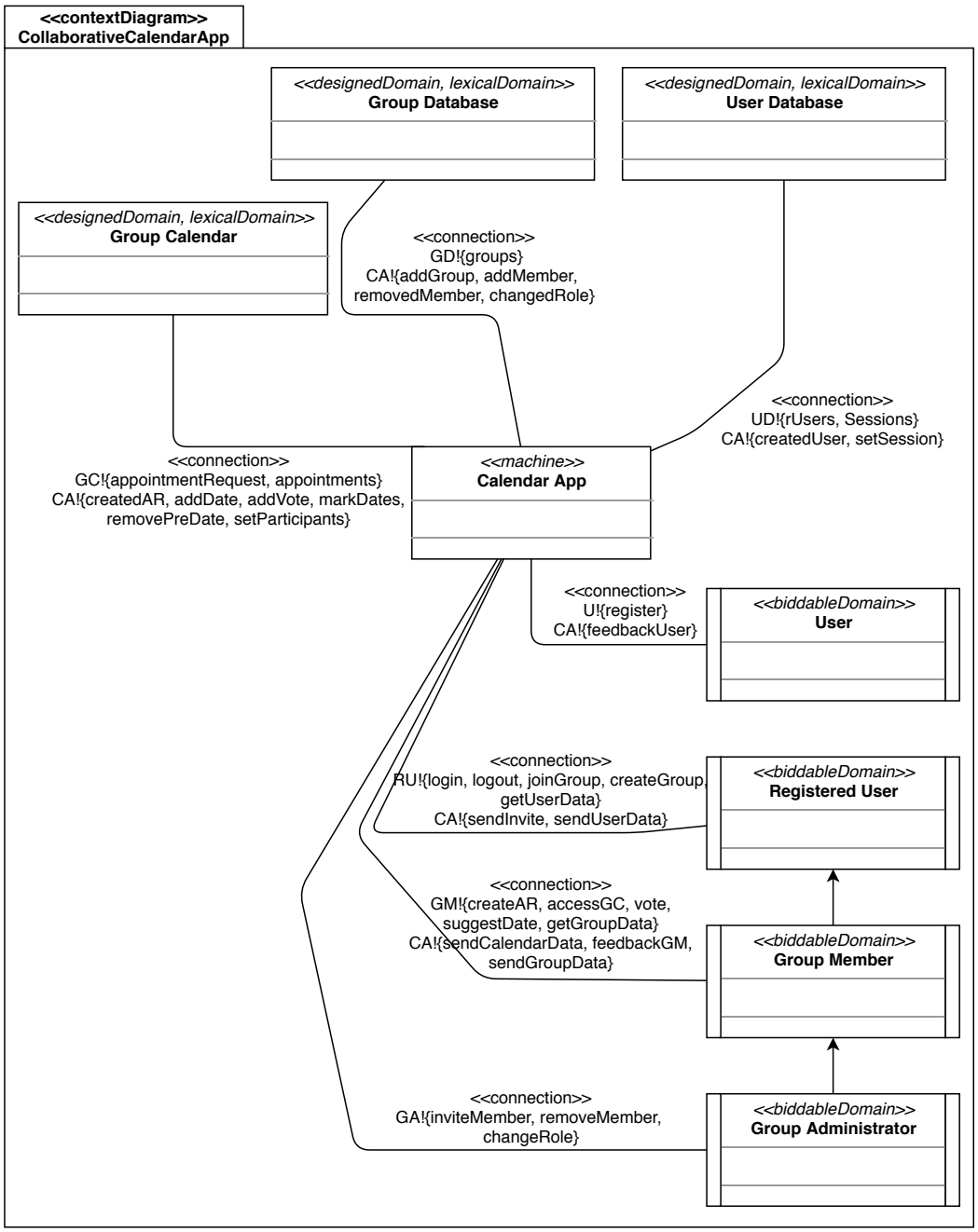

Figure 1.1: Contextdiagram

1.1.3 Validation

Validation I

The glossary contains the notions used in Requirements and Domain-Knowledge.
The notions mentioned in the Requirements and the Domain-Knowledge are contained in the glossary.

Validation II

Domains and phenomena of the context diagram must be consistent with Requirements and Domain-Knowledge.

Table 1.1: Validation

Notion in context diagram	**Notions in Requirements and Domain-Knowledge**	**type**
register	Users should be able to register	phenomenon
login	Users can login	phenomenon
logout	Users can log out	phenomenon
joinGroup	Registered users can join the groups they are invited to	phenomenon
createGroup	Registered users can create groups	phenomenon
sendInvite	*counterpart to inviteMember*	phenomenon
createAR	Group members can create a new appointment request	phenomenon
accessGC	All group members can access the group calendar	phenomenon
sendCalendarData	*counterpart to accessGC*	phenomenon
inviteMember	Group administrators can invite other registered users	phenomenon
removeMember	Group administrators can remove members from the group	phenomenon
changeRole	Group administrators can change a member's role	phenomenon
rUser	Registered Users	phenomenon
Sessions	*Necessary because of login/logout and setSession*	phenomenon
createUser	*counterpart to register*	phenomenon
setSession	*counterpart to login/logout*	phenomenon
groups	available groups	phenomenon
addGroup	*counterpart to createGroup*	phenomenon
addedMember	*counterpart to joinGroup*	phenomenon
removedMember	*counterpart to removeMember*	phenomenon
changedRole	*counterpart to changeRole*	phenomenon
appointmentRequest	open appointment requests	phenomenon
appointments	fixed dates	phenomenon
createdAR	*counterpart to createAR*	phenomenon
addDate	*counterpart to suggestDate*	phenomenon
addVote	*counterpart to vote*	phenomenon
markDates	marked as preliminary or fixed	phenomenon
removePreDate	All preliminary dates are removed from the calendar	phenomenon
vote	Planned participants have to select at least one of the existing dates	phenomenon

suggestDate	Planned Participants can suggest new dates for an appointment	phenomenon
setState	All planned participants [...] are recorded as actual participants	phenomenon
Calendar App	*The software we are going to build.*	domain
Group Calendar	calendar	domain
Group Database	*Necessary because of createGroup*	domain
User Database	*Necessary because of register*	domain
User	user	domain
Registered User	registered user	domain
Group Member	group members	domain
Group Administrator	group administrators	domain
Participant	planned participants and actual participants	domain

Validation III

There is only one context diagram given.

Validation IV

A context diagram has at least one machine domain.
Calendar App is one machine domain.

Table 1.2: Domain Validation

Domain	**Domain Type(s)**	**connected Domain(s)**	**connected Domain type(s)**
Group Calendar	designedDomain & lexicalDomain	Calendar App	machine Domain
Group Database	designedDomain & lexicalDomain	Calendar App	machine Domain
User Database	designedDomain & lexicalDomain	Calendar App	machine Domain
Calendar App	machine Domain	Group Calendar	designedDomain & lexicalDomain
		Group Database	designedDomain & lexicalDomain
		User Database	designedDomain & lexicalDomain
		Participants	biddableDomain
		User	biddableDomain
		Registered User	biddableDomain
		Group Member	biddableDomain
		Group Administrator	biddableDomain
Participants	biddableDomain	Calendar App	machine Domain
User	biddableDomain	Calendar App	machine Domain
Registered User	biddableDomain	Calendar App	machine Domain
Group Member	biddableDomain	Calendar App	machine Domain
Group Administrator	biddableDomain	Calendar App	machine Domain

Validation V

The machine domain must control at least one interface.
Calendar App controls several interfaces(feedbackGM, sendInvite, feedbackUser,sendUserData,...)

Validation VI

Biddable domains cannot be directly connected to lexical domains.
No biddable domain is connected to a lexical domain. (refer to table: 1.2)

Validation VII

Causal, designed, lexical, display, machine domain type are not allowed together with biddable domain.
Participants, User, Registered User, Group Member, Group Administrator are biddable domains only. (refer to table: 1.2)

Validation VIII

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

Table 1.3: Counterpart Validation

	biddable domain phenomena	**counterpart**
User		
	register	createdUser
Registered User		
	login, logout	setSession
	joinGroup	addedMember
	createGroup	addGroup
Group Member		
	createAR	createdAR
	accessGC	appointments, appointmentRequest
Group Administrator		
	inviteMember	addedMember
	removeMember	removedMember
	changeRole	changedRole
Participant		
	vote	addVote
	suggestDate	addDate

Validation IX

Connection domains must have at least one observed and one controlled interface.
Context diagram contains no connection domain.

Validation X

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.
Context diagram contains no connection domain.

Validation XI

For each pheonomenon 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

We can derive the following problem diagrams

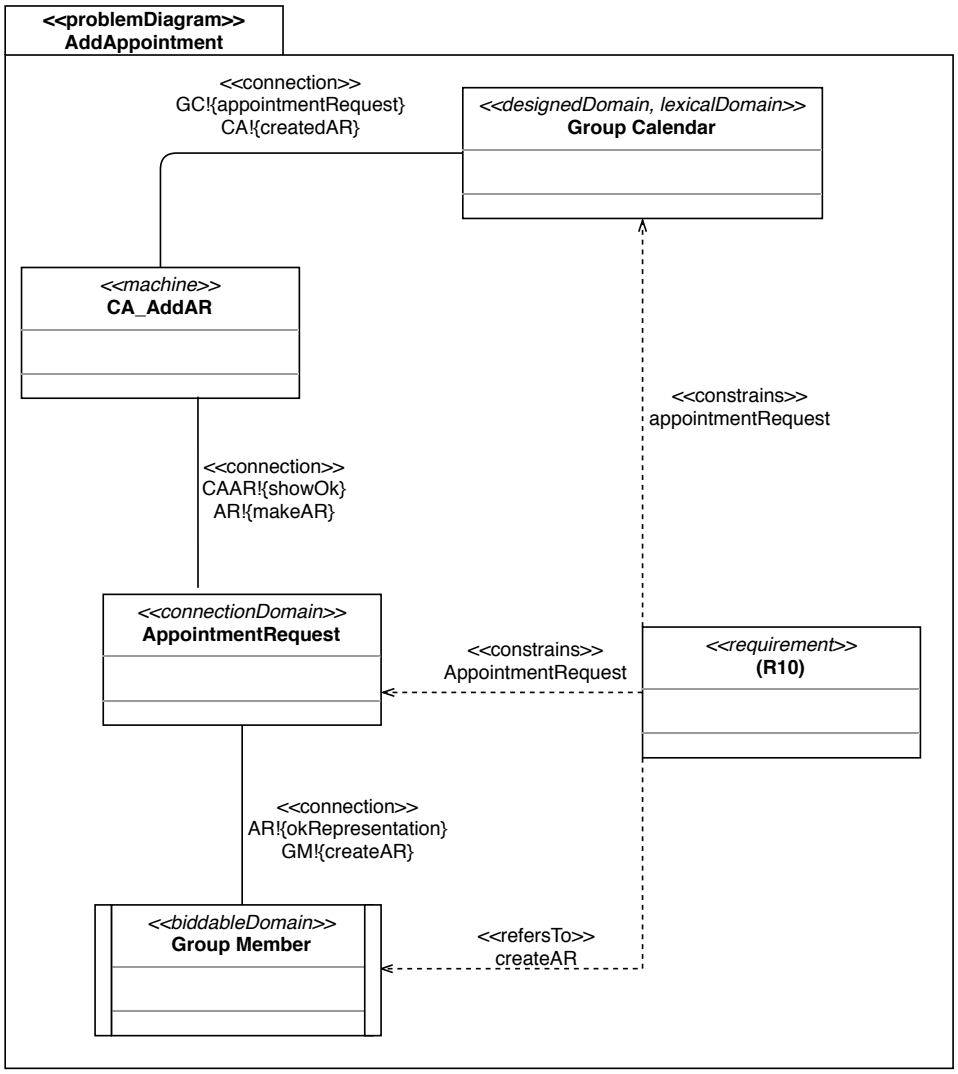

Figure 1.2: Problem diagram for R10

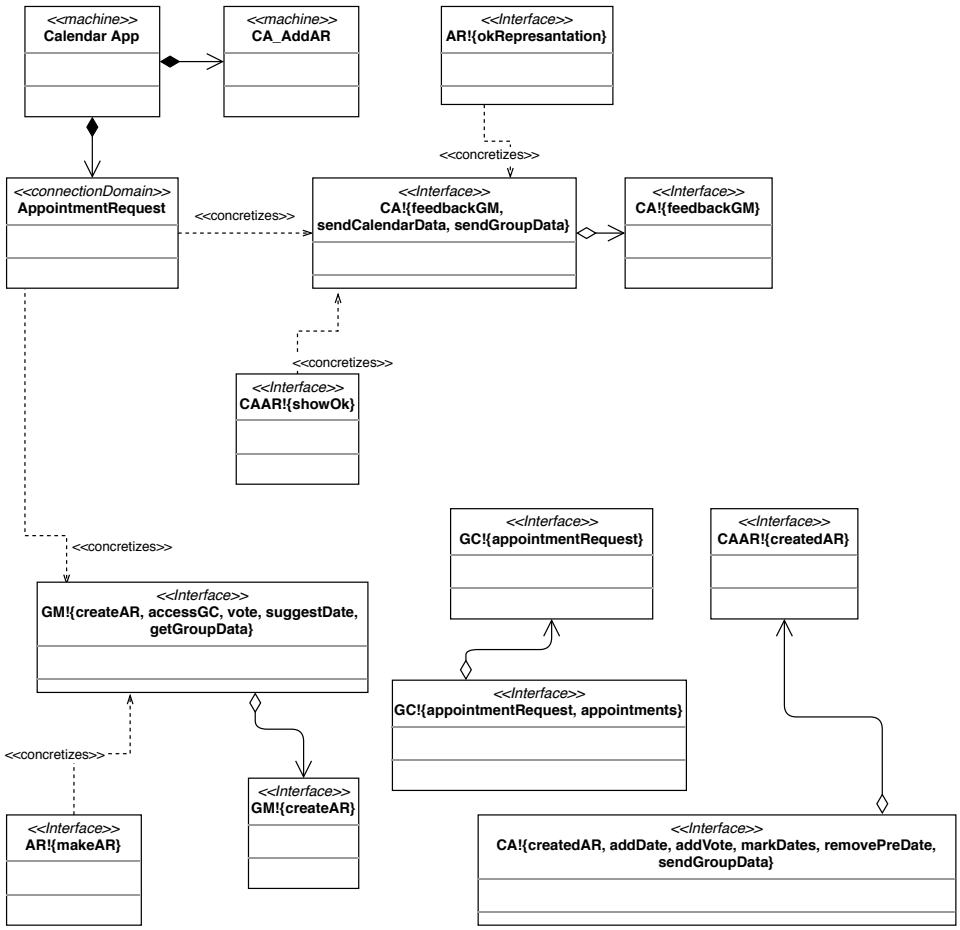

Figure 1.3: Mapping for R10

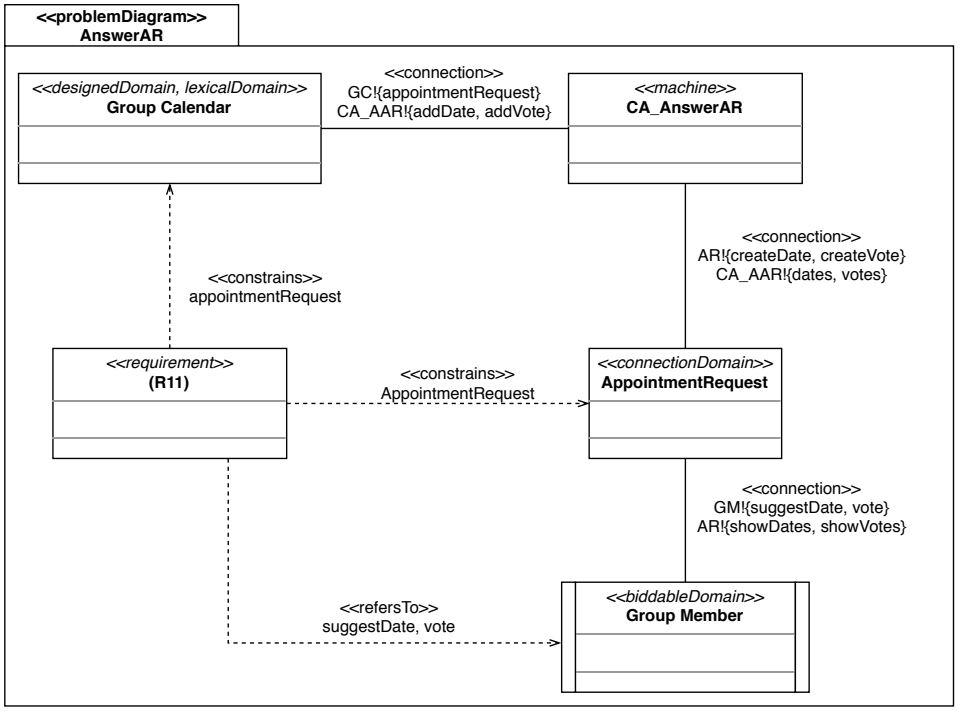

Figure 1.4: Problem diagram for R11

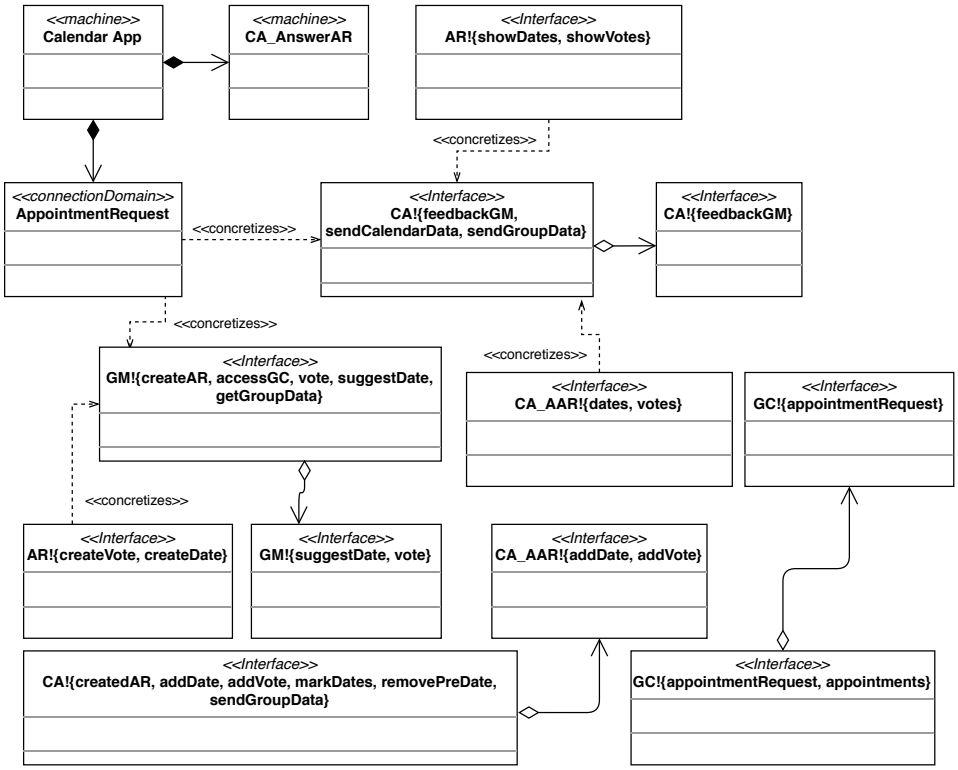

Figure 1.5: Mapping for R11

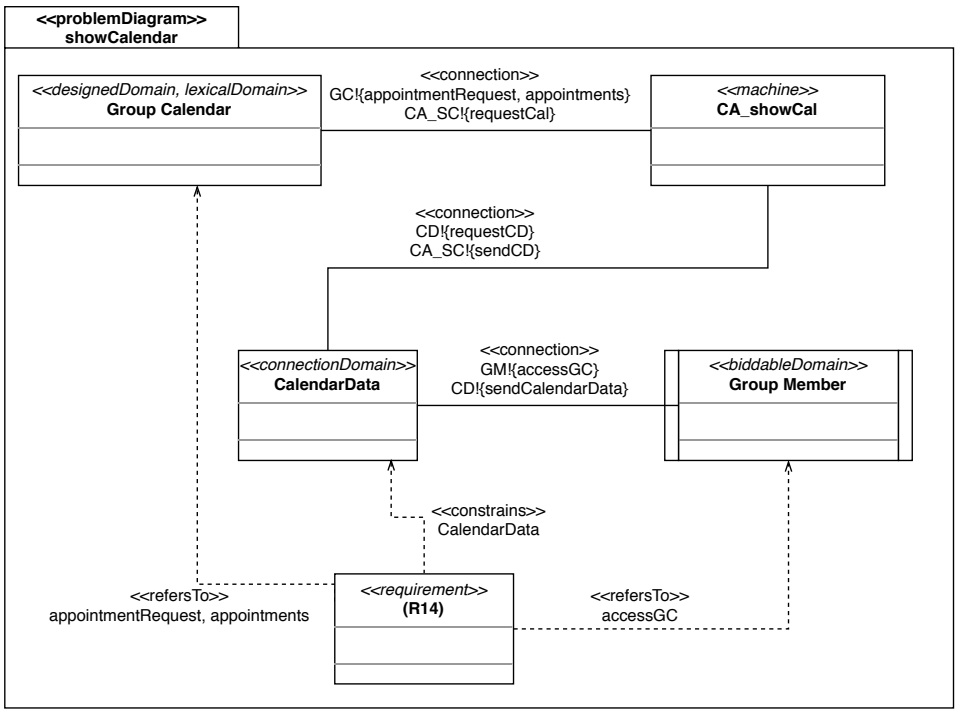

Figure 1.6: Problem diagram for R14

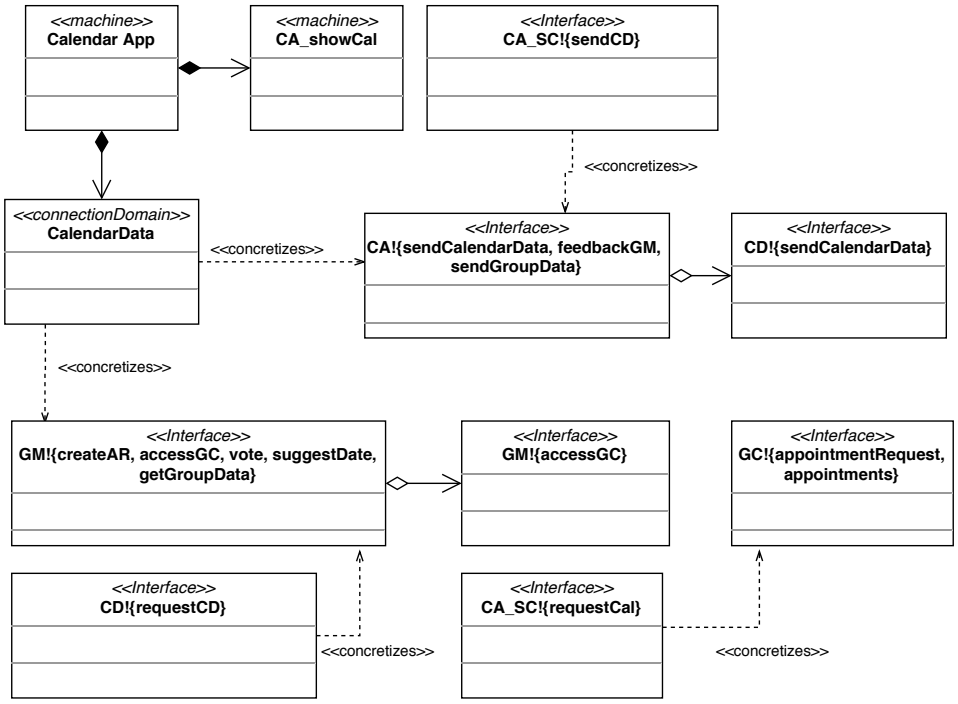

Figure 1.7: Mapping for R14

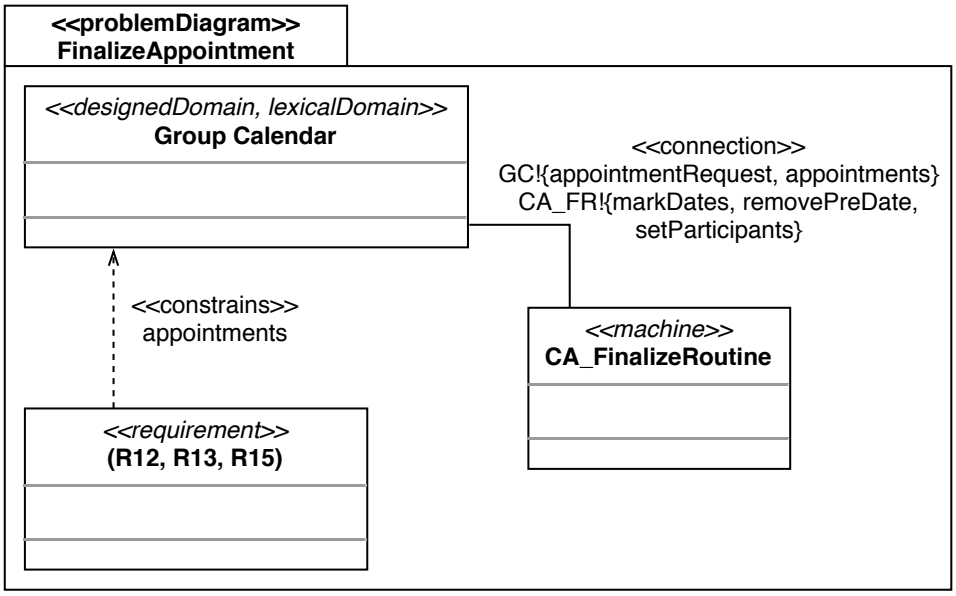

Figure 1.8: Problem diagram for R12, R13, R15

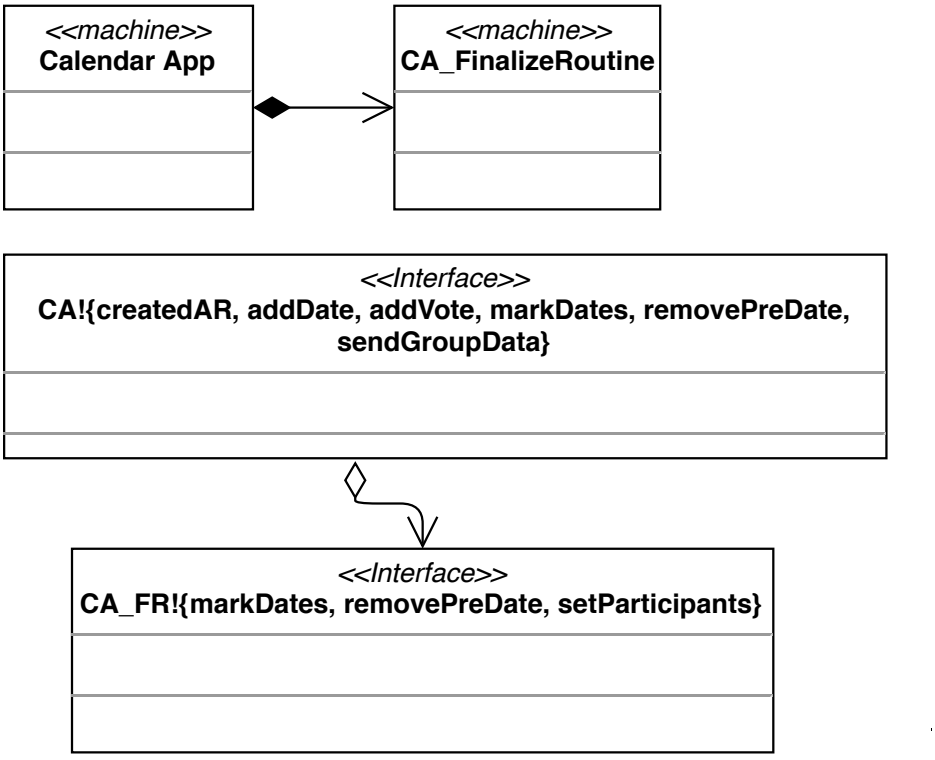

Figure 1.9: Mapping for R12, R13, R15

1.2.2 Problem frames

The Problem diagram for AddAppointment fits to the Problem frame update II
The Problem diagram for AnswerAR fits to the Problem frame update II
The Problem diagram for showCalendar fits to the Problem frame query II
The Problem diagram for FinalizeAppointment fits to the Problem frame simple transformation

1.2.3 Validation

Validation I

The required requirements R are covered in some subproblem

Table 1.4: A2 Validation I

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R10	Add-Appointment	CA_AddAR	machine		createdAR, showOk
		Appointment-Request	connection	X	makeAR, okRepresentation
		GroupMember	biddable		createAR
		Group-Calendar	designed, lexical	X	appointment-Request
R11	AnswerAR	CA_AnswerAR	machine		dates, votes
		Group Calendar	designed, lexical	X	appointment-Request
		Appointment-Request	connection	X	createDate, createVote, showDates, showVotes
		Group Member	biddable		suggestDate, vote
R14	showCalendar	CA_showCal	machine		requestCal, sendCD
		CalendarData	connection	X	requestCD, sendCalendarData
		GroupMember	biddable		accessGC
		Group-Calendar	designed, lexical		appointment-Request, appointments
R12, R13, R15	Finalize-Appointment	CA_Finalize-Routine	machine		markDates, removePreDate, setParticipants
		Group Calendar	designed, lexical	X	appointmentRequest, appointments

Our task does not include the creation of problem diagrams for the other requirements.

Validation II

A problem diagram has exactly one machine domain

Table 1.5: A2 Validation II

requiremen	covered in	contained	domain type	constrained	controlled phenom-ena
R10	Add-Appointment	CA_AddAR Appointment-Request GroupMember Group-Calendar	machine connection biddable designed, lexical	 X X	createdAR, showOk makeAR, okRepresen-tation createAR appointment-Request
R11	AnswerAR	CA_AnswerAR Group Calen-dar Appointment-Request Group Mem-ber	machine designed, lexical connection biddable	 X X	dates, votes appointment-Request createDate, createVote, showDates, showVotes suggestDate, vote
R14	showCalendar	CA_showCal CalendarData GroupMember Group-Calendar	machine connection biddable designed, lexical	 X	requestCal, sendCD requestCD, sendCalen-darData accessGC appointment-Request, ap-pointments
R12, R13, R15	Finalize-Appointment	CA_Finalize-Routine Group Calen-dar	machine designed do-main, lexical domain	 X	markDates, removePre-Date, set-Participants appointmen-tRequest, appoint-ments

Validation III

A problem diagram contains at least one requirement

Table 1.6: A2 Validation III

requiremen	covered in	contained domain	domain type	constrained	controlled phenom-ena
R10	Add-Appointment	CA_AddAR	machine		createdAR, showOk
		Appointment-Request	connection	X	makeAR, okRepresen-tation
		GroupMember	biddable		createAR
		Group-Calendar	designed, lexical	X	appointment-Request
R11	AnswerAR	CA_AnswerAR	machine		dates, votes
		Group Calen-dar	designed, lexical	X	appointment-Request
		Appointment-Request	connection	X	createDate, createVote, showDates, showVotes
		Group Mem-ber	biddable		suggestDate, vote
R14	showCalendar	CA_showCal	machine		requestCal, sendCD
		CalendarData	connection	X	requestCD, sendCalen-darData
		GroupMember	biddable		accessGC
		Group-Calendar	designed, lexical		appointment-Request, ap-pointments
R12, R13, R15	Finalize-Appointment	CA_Finalize-Routine	machine		markDates, removePre-Date, set-Participants
		Group Calen-dar	designed, lexical	X	appointmen-tRequest, appoint-ments

Validation IV

The machine domain must control at least one interface

Table 1.7: A2 Validation IV

requiremen	covered in	contained	domain type	constrained	controlled phenom-ena
R10	Add-Appointment	CA_AddAR	machine		createdAR, showOk
		Appointment-Request	connection	X	makeAR, okRepresen-tation
		GroupMember Group-Calendar	biddable designed, lexical	X	createAR appointment-Request
R11	AnswerAR	CA_AnswerAR Group Calen-dar	machine designed, lexical	X	dates, votes appointment-Request
		Appointment-Request	connection	X	createDate, createVote, showDates, showVotes
		Group Mem-ber	biddable		suggestDate, vote
R14	showCalendar	CA_showCal	machine		requestCal, sendCD
		CalendarData	connection	X	requestCD, sendCalen-darData
		GroupMember Group-Calendar	biddable designed, lexical		accessGC appointment-Request, ap-pointments
R12, R13, R15	Finalize-Appointment	CA_Finalize-Routine	machine		markDates, removePre-Date, set-Participants
		Group Calen-dar	designed, lexical	X	appointmen-tRequest, appoint-ments

Validation V

Requirements constrain at least one domain

Table 1.8: A2 Validation V

requiremen	covered in	contained domain	domain type	constrained	controlled phenomena
R10	Add-Appointment	CA_AddAR	machine		createdAR, showOk
		Appointment-Request	connection	X	makeAR, okRepresentation
		GroupMember	biddable		createAR
		Group-Calendar	designed, lexical	X	appointment-Request
R11	AnswerAR	CA_AnswerAR	machine		dates, votes
		Group Calendar	designed, lexical	X	appointment-Request
		Appointment-Request	connection	X	createDate, createVote, showDates, showVotes
		Group Member	biddable domain		suggestDate, vote
R14	showCalendar	CA_showCal	machine		requestCal, sendCD
		CalendarData	connection	X	requestCD, sendCalendarData
		GroupMember	biddable		accessGC
		Group-Calendar	designed, lexical		appointment-Request, appointments
R12, R13, R15	Finalize-Appointment	CA_Finalize-Routine	machine		markDates, removePreDate, setParticipants
		Group Calendar	designed, lexical	X	appointmentRequest, appointments

Validation VI

Requirements do not constrain machine(s)

Table 1.9: A2 Validation VII

requirement	covered in	contained domain	domain type	constrained	controlled phenomena
R10	Add-Appointment	CA_AddAR	machine		createdAR, showOk makeAR, okRepresentation
		Appointment-Request	connection	X	createAR appointment-Request
		GroupMember Group-Calendar	biddable designed, lexical	X	
R11	AnswerAR	CA_AnswerAR Group Calendar	machine designed, lexical	X	dates, votes appointment-Request createDate, createVote, showDates, showVotes suggestDate, vote
		Appointment-Request	connection	X	
		Group Member	biddable		
R14	showCalendar	CA_showCal	machine		requestCal, sendCD requestCD, sendCalendarData accessGC appointment-Request, appointments
		CalendarData	connection	X	
		GroupMember Group-Calendar	biddable designed, lexical		
R12, R13, R15	Finalize-Appointment	CA_Finalize-Routine	machine		markDates, removePreDate, setParticipants appointmentRequest, appointments
		Group Calendar	designed, lexical	X	

Validation VII

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

Table 1.10: A2 Validation VII

requiremen	covered in	contained domain	domain type	constrained	controlled phenom-ena
R10	Add-Appointment	CA_AddAR	machine		createdAR, showOk makeAR, okRepresen-tation
		Appointment-Request	connection	X	createAR
		GroupMember	biddable		appointment-Request
		Group-Calendar	designed, lexical	X	
R11	AnswerAR	CA_AnswerAR	machine		dates, votes
		Group Calen-dar	designed, lexical	X	appointment-Request
		Appointment-Request	connection	X	createDate, createVote, showDates, showVotes
		Group Mem-ber	biddable		suggestDate, vote
R14	showCalendar	CA_showCal	machine		requestCal, sendCD
		CalendarData	connection	X	requestCD, sendCalen-darData
		GroupMember	biddable		accessGC
		Group-Calendar	designed, lexical		appointment-Request, ap-pointments
R12, R13, R15	Finalize-Appointment	CA_Finalize-Routine	machine		markDates, removePre-Date, set-Participants
		Group Calen-dar	designed, lexical	X	appointmen-tRequest, appoint-ments

Validation VIII

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

Table 1.11: A2 Validation VIII

connection Domain	phenomenom controlled by connection domain	connected Domain	phenomenon controlled by connected domain
AppointmentRequest	makeAR	GroupMember	createAR
	okRepresentation	CA_AddAR	showOk
	createDate	GroupMember	suggestDate
	createVote	GroupMember	vote
	showDates	CA_AnswerAR	dates
	showVotes	CA_AnswerAR	votes
CalendarData	requestCD	GroupMember	accessGC
	sendCalendarData	CA_showCal	sendCD

Validation IX

For each phenomenon controlled by a connection domain, there must be at least one phenomenon controlled by one of the connection domains

Table 1.12: A2 Validation IX

connection domain	phenomenom controlled by connection domain	connected Domain	phenomenom controlled by connected domain
AppointmentRequest	makeAR	GroupMember	createAR
	okRepresentation	CA_AddAR	showOk
	createDate	GroupMember	suggestDate
	createVote	GroupMember	vote
	showDates	CA_AnswerAR	dates
	showVotes	CA_AnswerAR	votes
CalendarData	requestCD	GroupMember	accessGC
	sendCalendarData	CA_showCal	sendCD

Validation X

For each phenomenon observed by a connection domain, there must be at least one phenomenon controlled by the connection domain

Table 1.13: A2 Validation X

connection domain	phenomenom observed by connection domain	phenomenom controlled by connection domain
AppointmentRequest	createAR	makeAR
	showOk	okRepresentation
	suggestDate	createDate
	vote	createVote
	dates	showDates

connection domain	phenomenom observed by connection domain	phenomenom controlled by connection domain
	votes	showVotes
CalendarData	accessGC	requestGC
	sendCD	sendCalendarData

Validation XI

The problem diagrams must be consistent to the context diagram.
Refer to the provided mappings

Validation XII

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

Table 1.14: A2 Validation XII

problem diagram	opertor	related domains or phenomena
AddAppointment	leave out domain	User, Registered User, Group Administrator, Group Database, User Database
	introduce connection/display domain	AppointmentRequest
	split interface	GM!{...},GC!{...},CA!{...}
	concretize interface	CA!{...},GM!{...}
AnswerAR	leave out domain	User, Registered User, Group Administrator, Group Database, User Database
	introduce connection/display domain	AppointmentRequest
	split interface	GC!{...},CA!{...}
	concretize interface	CA!{...},GM!{...}
showCalendar	leave out domain	User, Registered User, Group Administrator, Group Database, User Database
	introduce connection/display domain	CalendarData
	split interface	GM!{...}
	concretize interface	CA!{...},GM!{...}
FinalizeAppointment	leave out domain	User, Registered User, Group Member, Group Administrator, Group Database, User Database
	introduce connection/display domain	
	split interface	CA!{...}
	concretize interface	CA!{...}

Validation XIII

All connections in a problem diagram correspond to a connection in the frame diagram

Table 1.15: A2 Validation XIII

problem diagram	problem frame	connections in pd	connection in pf	domain type 1	domain type 2
Add-Appointment	update2	CA!{createdAR}, GC!{appointmentRequest}	DB!Y1, UM!Y2	machine	lexicalDomain
		CAAR!{showOk}, AR!{makeAR}	UM!E4, IOD!E8	machine	connection-Domain
		AR!{okRepresentation}, GM!{createAR}	UO!E6, IOD!C7	biddable Domain	Connection Domain
AnswerAR	update2	GC!{appointmentRequest}, CA_AAR!{addDate,addVote}	DB!Y1, UM!Y2	machine	lexicalDomain
		AR!{createDate, create-Vote}, CA_AAR{dates, votes}	UO!E6, IOD!C7	machine	connection-Domain
		GM!{suggestDate, vote}, AR!{showDates, showVotes}	UO!E6, IOD!C7	biddable Domain	Connection Domain
showCalendar	query2	GC!{appointments, appointmentRequest}	DB!Y1	machine	lexical domain
		CD!{requestCD}, CA_SC!{sendCD}	QM!Y1, IOD!C6	machine	connection domain
		GM!{accessGC},CD!{sendCalendarData}	IOD!E7, EO!E5	biddable domain	connection domain
Finalize-Appointment	simple transformation	GC!{appointmentRequest, appointments}, CA_FR!{markDates, removePreDate, setParticipants}	STM!Y1, W!Y2	machine	lexical domain

Validation XIV

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

Table 1.16: Validation XIV

problem diagram	problem frame	constrained domains in pd	constrains domains in pf	domain type
AddAppointment	update II	Group Calendar	Data Base	DesignedDomain, LexicalDomain
		AppointmentRequest	Input Output Device	ConnectedDomain
AnswerAR	update II	Group Calendar	Data Base	DesignedDomain, LexicalDomain
		AppointmentRequest	Input Output Device	ConnectedDomain
showCalendar	query II	Group Calendar	Data Base	DesignedDomain, LexicalDomain
		CalendarData	Input Output Device	ConnectedDomain
FinalizeAppointment	simple transformation	Group Calendar	Workpieces	DesignedDomain, LexicalDomain

Validation XV

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

Table 1.17: Validation XV

problem diagram	problem frame	constrained domains in pd	constrains domains in pf	domain type
AddAppointment	update II	Group Member	Update Operator	BiddableDomain
AnswerAR	update II	Group Member	Update Operator	BiddableDomain
showCalendar	query II	Group Member	Enquiry Operator	BiddableDomain
		Group Calendar	Data Base	DesignedDomain, LexicalDomain

1.3 A3

1.3.1 R10

Using the domain knowledge A1, A2, A6, A7 and F3 we can derive the specifications:
(S10a) AppointmentRequest: When AppointmentRequest receives the command "createAR", then the command is forwarded to the machine with the command "makeAR". The results are received via the command "showOk" and shown to the group member via the command "okRepresentation".
(S10b) CA_AddAR: When the machine receives the command "makeAR", then the command is forwarded to the database via the command "createdAR". As answer it receives the data "appointmentRequest" and forwarded this via the command "showOK".
(S10c) GroupCalendar: When the database receives the command "createdAR", it creates the new appointment request in the calendar and gives as feedback the data "appointmentRequest" with all actual requests. Correctness condition:
$(S10a) \wedge (S10b) \wedge (S10c) \wedge A1 \wedge A2 \wedge A6 \wedge A7 \wedge F3 \implies R10$

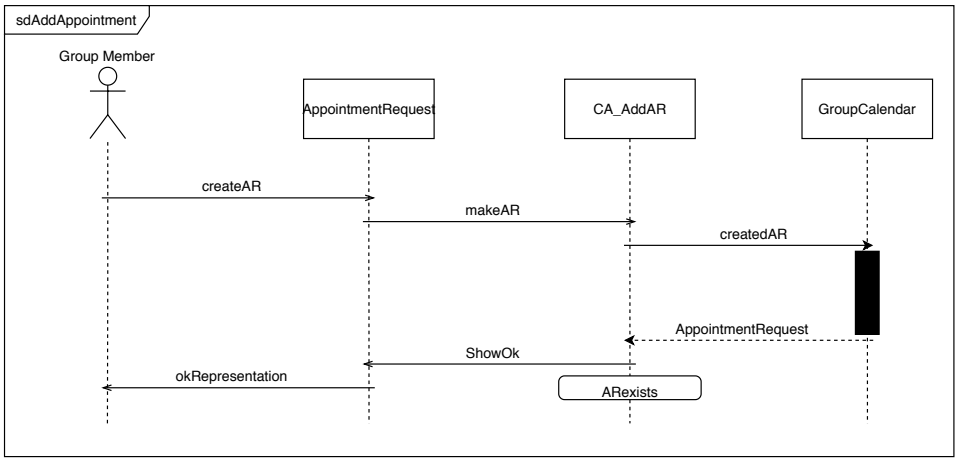

Figure 1.10: Sequenzdiagram R10, sdAddAppointment

1.3.2 R11

Using the domain knowledge A1, A2, A5, A6, A7, A8, A10, A11 and F3 we can derive the specifications:
(S11a) AppointmentRequest: When the AppointmentRequest receives the command "suggestDate", then the command is forwarded to the machine with the command "createDate". It receives feedback via the command "dates" and gives this to the GroupMember via "showDates". When the AppointmentRequest receives the command "vote", then the command is forwarded to the machine with the command "createVote". It receives feedback via the command "votes" and gives this to the GroupMember via "showVotes".
(S11b) CA_AnswerAR: When the machine receives the command "createDate", then the GroupCalendar is supposed to add a new date to the database with the command "addDate". The machine receives the data "appointmentRequest" with all current AppointmentRequests. The machine gives feedback via the command "dates". When the machine receives the command "createVote", then this command is forwarded to the GroupCalendar via the command "addVote". It receives the data "appointmentRequest"and transfers this via the command "votes" to AppointmentRequest.
(S11c) GroupCalendar: When the GroupCalendar receives the command "addDate", a new date will be added to the calendar. When it receives the command "addVote", a new vote for the AppointmnetRequest ist added in the calendar. The results are given with the data "appointmentRequest".

Correctness condition:
$(S11a) \wedge (S11b) \wedge (S11c) \wedge A1 \wedge A2 \wedge A5 \wedge A6 \wedge A7 \wedge A8 \wedge A10 \wedge A11 \wedge F3 \implies R11$

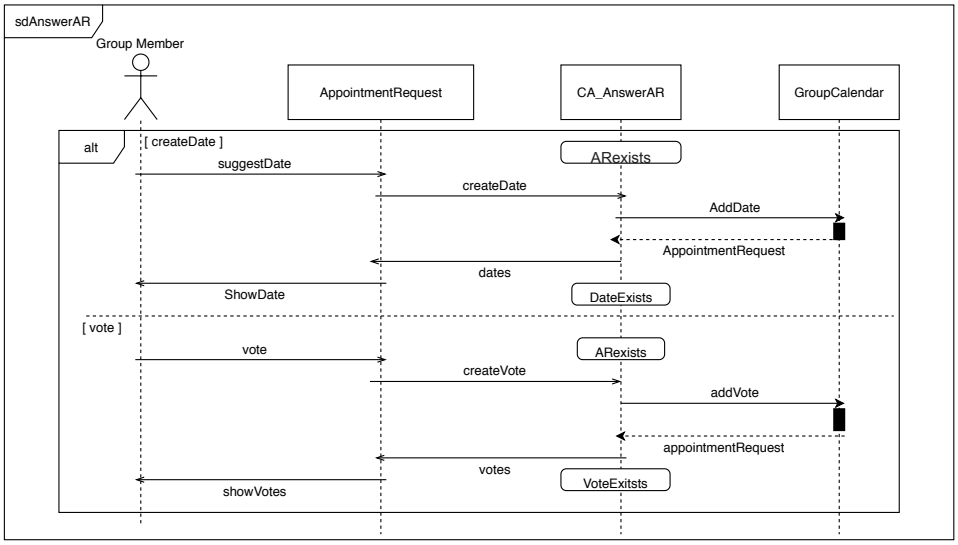

Figure 1.11: Sequenzdiagram R11, sdAnswerAR

1.3.3 R14

Using the domain knowledge A1, A2 and F3 we can derive the specifications:
(S14a) CalendarData: When CalendarData receives the command "accessGC", then the command is forwarded to the machine with the command "requestCD". The results are received via the command "sendCD" and shown to the group member via the command "sendCalendarData".
(S14b) CA_showCal: When the machine receives the command "requestCD",it requests all dates in the calendar with the command "requestCal". The result is received as the data "appointmentRequest" and "appointments". This is forwarded to CalendarData with the command "sendCD".
(S14c) GroupCalendar: When the GroupCalendar receives the command "requestCal", all dates in the calendar are returned as the data "appointmentRequest" and "appointments".
Correctness condition:
$(S14a) \wedge (S14b) \wedge (S14c) \wedge A1 \wedge A2 \wedge F3 \implies R14$

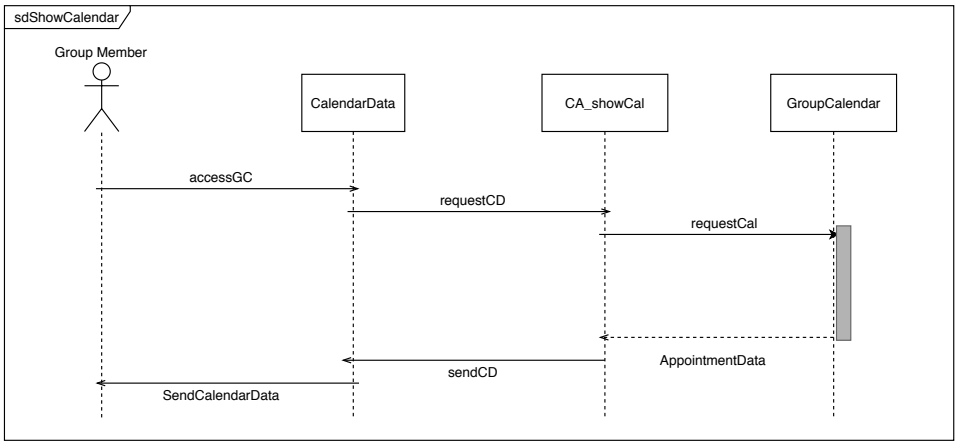

Figure 1.12: Sequenzdiagram R14, sdShowCalendar

1.3.4 R12, R13, R15

Using the domain knowledge A6, A7, A9 and F3 we can derive the specifications:
(S12/13/15a) CA_FinalizeRoutine: When the machine receives the command "checkDeadline" and the deadline is reached, then the commands "markDates", "removePreDate" and "setParticipants" are forwarded to the GroupCalendar to find a fixed date for the appointment. The results are recieved as data "appointmentRequest" and "appointments".
(S12/13/15b) GroupCalendar: When the GroupCalendar receives the commands "markDates", "removePreDate" and "setParticipants",it removes the preliminary dates and sets the participants for the appointment. It returns the data "appointmentRequest" and "appointments".
Correctness condition:
$(S12/13/15a) \wedge (S12/13/15b) \wedge A6 \wedge A7 \wedge A9 \wedge F3 \implies (R12, R13, R15)$

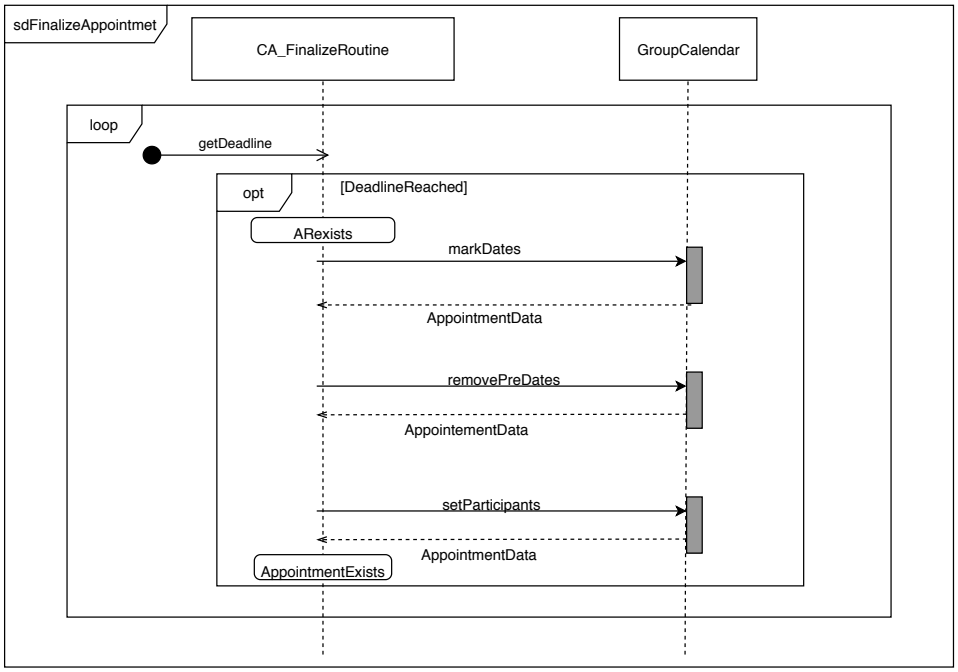

Figure 1.13: Sequenzdiagram R12, R13, R15, sdFinalizeappointment

1.3.5 Validation

Validation I

$S(abstract) \wedge D$ are non - contradictory.
No contradictions can be found in $S(abstract) \wedge D$.

$S(abstract) \wedge D \implies R$.
$(S10a) \wedge (S10b) \wedge (S10c) \wedge A2 \wedge A6 \wedge A7 \wedge F3 \implies R10$
$(S11a) \wedge (S11b) \wedge (S11c) \wedge A2 \wedge A5 \wedge A6 \wedge A7 \wedge A8 \wedge A10 \wedge A11 \implies R11$
$(S12/13/15a) \wedge (S12/13/15b) \wedge A9 \implies (R12, R13, R15)$

Validation II

Messages and phenomena are consistent.

message in scenario	source	target	phenomena in problem diagram
createAR	GroupMember	AppointmentRequest	GM!{createAR}
makeAR	AppointmentRequest	CA_AddAR	AR!{makeAR}
createdAR	CA_AddAR	GroupCalendar	CA!{createdAR}
showOK	CA_AddAR	AppointmentRequest	CA!{showOK}
okRepresentation	AppointmentRequest	GroupMember	AR!{okRepresentation}
suggestDate	GroupMember	AppointmentRequest	GM!{suggestDate}
createDate	AppointmentRequest	CA_AnswerAR	AR!{createDate}
addDate	CA_AnswerAR	GroupCalendar	CA_AAR!{addDate}
dates	CA_AnswerAR	AppointmentRequest	CA_AAR!{dates}
showDate	AppointmentRequest	GroupMember	AR!{showDate}
vote	GroupMember	AppointmentRequest	GM!{vote}
createVote	AppointmentRequest	CA_AnswerAR	AR!{createVote}
addVote	CA_AnswerAR	GroupCalendar	CA_AAR!{addVote}
votes	CA_AnswerAR	AppointmentRequest	CA_AAR!{votes}
showVotes	AppointmentRequest	GroupMember	AR!{showVotes}
accessGC	GroupMember	CalnedarData	GM!{accessGC}
requestCD	CalendarData	CA_showCal	CD!{requestCD}
requestCal	CA_showCal	GroupCalendar	CA_SC!{requestCal}
sendCD	CA_showCal	CalendarData	CA_SC!{sendCD}
sendCalendarData	CalendarData	GroupMember	CD!{sendCalendarData}
markDates	CA_FinalizeRoutine	GroupCalendar	CA_FR!{markDates}
removesPreDate	CA_FinalizeRoutine	GroupCalendar	CA_FR!{removePreDate}
setParticipants	CA_FinalizeRoutine	GroupCalendar	CA_FR!{setParticipants}

Validation III

Lexical domains are not sources of messages

message in scenario	source	domain type
createAR	GroupMember	BiddableDomain
makeAR	AppointmentRequest	ConnectionDomain
createdAR	CA_AddAR	machine domain
showOk	CA_AddAR	machine domain
okRepresentation	AppointmentRequest	ConnectionDomain
suggestDate	GroupMember	BiddableDomain
createDate	AppointmentRequest	ConnectionDomain
addDate	CA_AnswerAR	machine domain
dates	CA_AnswerAR	machine domain
createVote	AppointmentRequest	ConnectionDomain
addVote	CA_AnswerAR	machine domain
votes	CA_AnswerAR	machne domain
showVotes	AppointmentRequest	ConnectionDomain
accessGC	GroupMember	BiddableDomain
requestCD	CalendarData	ConnectionDomain
requestCal	CA_showCal	machine domain

message in scenario	source	domain type
sendCD	CA_showCal	machine domain
sendCalendarData	CalendarData	ConnectionDomain
markDates	CA_FinalizeRoutine	machine domain
removePreDate	CA_FinalizeRoutine	machine domain
setParticipants	CA_FinalizeRoutine	machine domain

Validation IV

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

subproblem	normal case	exceptional case
AddAppointment	sdAddAppointment	
AnswerAR	sdAnswerAR	
ShowCalendar	sdShowCalendar	
FinalizeAppointment	sdFinalizeAppointment	

1.4 A4

1.4.1 Technical Context Diagram

Technical realization of domains from context and problem diagrams:

AppointmentRequest: Realized using Apache Tomcat as server platform and GroupMemberWebBrowser(browser of Group Member).

CalendarData: Realized using Apache Tomcat as server platform and GroupMemberWebBrowser (browser of Group Member).

GroupCalendar: Realized as SQLDatabase on the same computer as the machine. Therefore, the database is connected by a call-and-return interface and used with SQL commands.

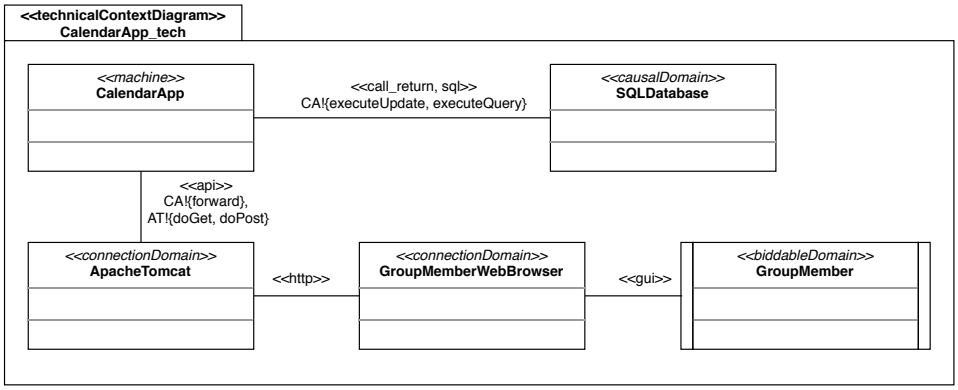

Figure 1.14: TechnicalContextDiagram

Mapping diagram for technical context diagram.

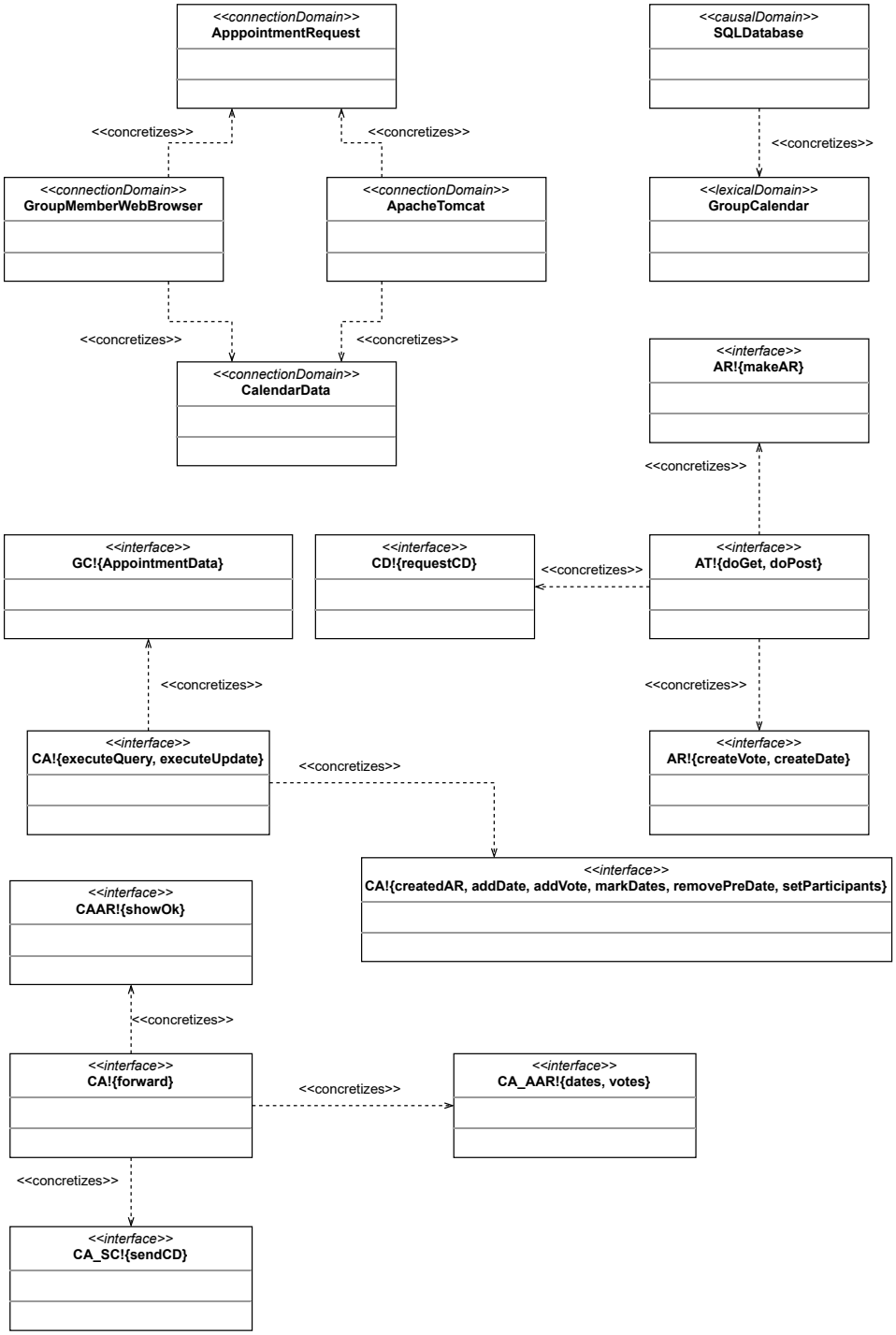

Figure 1.15: Mapping TechnicalContextDiagram

Technical interfaces of the machine:

SQL commands: defined in FIPS PUB 127-2(U.S. DEPARTMENT OF COMMERCE/ National Institute of Standards and Technology, 1993)
Operations executeQuery and executeUpdate are defined in interface java.sql.Statement (https://docs.oracle.com/javase/8/docs/api/index.html?java/sql/Statement.html)

API for ApacheTomcat(http://tomcat.apache.org/tomcat-9.0-doc/index.html):
Operations doGet and doPost are defined in abstract class javax.servlet.http.HttpServlet (https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServlet.html)
Operation forward defined in interface javax.servlet.RequestDispatcher (http://docs.oracle.com/javaee/7/api/javax/servlet/RequestDispatcher.html)

Technical interfaces in the environment:

HTTP(Hypertext Transfer Protocal): defined in RFC 2616, (Network Working Group, 1999)
GUI: User interfaces of HTML webpages (defined by https://www.w3.org/TR/html5/) presented by GroupMemberWebBrowser.

1.4.2 Validation

Validation I

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

Table 1.21: A4 Validation I

Message	new phenomena and domains
createAR	ApacheTomcat, HTTP
suggestDate	ApacheTomcat, HTTP
vote	ApacheTomcat, HTTP
accessGC	ApacheTomcat, HTTP

An internal message can be realized using SQL commands.

Validation II

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:
See the provided mapping diagram.

Validation III

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

Table 1.22: A4 Validation III

Problem Diagram	Domain connected with the machine	Element in the TCD
AddAppointment	GroupCalendar	SQLDatabase
	AppointmentRequest	GroupMemberWebBrowser, ApacheTomcat
AnswerAR	GroupCalendar	SQLDatabase
	AppointmentRequest	GroupMemberWebBrowser, ApacheTomcat
showCalendar	GroupCalendar	SQLDatabase
	CalendarData	GroupMemberWebBrowser, ApacheTomcat
FinalizeAppointment	GroupCalendar	SQLDatabase

1.5 A5

1.5.1 AddAppointment

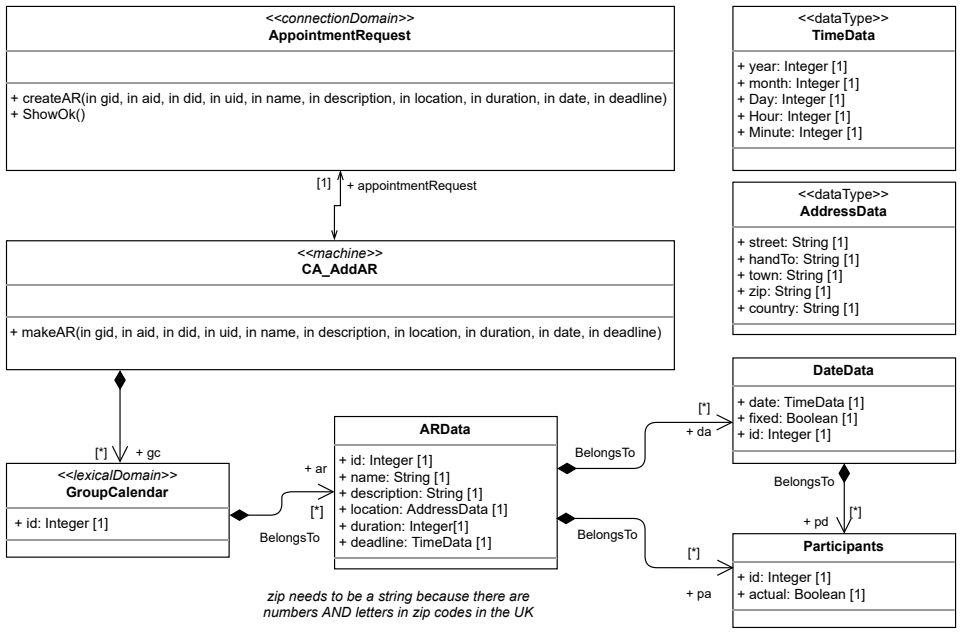

Figure 1.16: Class model: AddAppointment

Name: makeAR
Description: An appointment request is added to the GroupCalendar.
OCL constraint:

```
context CA_AddAR::makeAR(gid:Integer, aid:Integer, did:Integer,
    uid:Integer, name: String, description: String, location:
    AddressData, duration: Integer, date: TimeData, deadline:
    Integer)
pre: gc->one(gp: GroupCalendar | gp.id=gid)
post: let g: GroupCalendar = gc->any(gr: GroupCalendar | gr.id=gid
    ) in
g.ar->one(ap:ARData|ap.id=aid and ap.name=name and ap.description=
    description and ap.location=location and ap.duration=duration
    and ap.pa->forAll(p: Participants|p.id=uid)=true and ap.da->
    forAll(d: DateData|d.id=did, d.date=date)=true and ap.deadline=
    deadline) and
g.ar->size() = g.ar@pre->size()+1 and
AppointmentRequest^showOk()
```

Group calendars have a unique id.
OCL constraint:

```
context: GroupCalendar
inv: GroupCalendar.allInstances()->isUnique(id)
```

Appointment requests have a unique id.
OCL constraint:

```
context: ARData
inv: ARData.allInstances()->isUnique(id)
```

Participants have a unique id.
OCL constraint:

```
context: Participants
inv: Participants.allInstances()->isUnique(id)
```

Dates have a unique id.
OCL constraint:

```
context: DateData
inv: DateData.allInstances()->isUnique(id)
```

Name: createAR
Description: Forwards an appointment request from a group member to the machine.
OCL constraint:

```
context: AppointmentRequest::createAR(gid: Integer, aid: Integer,
    did: Integer, uid: Integer, name: String, description: String,
    location: AddressData, duration: Integer, date: Integer,
    deadline: Integer)
pre: true
post: CA_AddAR^makeAR(gid, aid, did, uid, name, description,
    location, duration, date, deadline)
```

Validation

Operation specifications must be consitent with abstract specifications:
The operation specification of makeAR 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 are covered in the postcondition.

Parameters must be used in the pre- and/or postcondition:
The parameters are used in the postcondition.

All parameters of operation must be known by the caller and all parameters of sent messages must be known by the machine:
GroupMember can inout all parameters to AppointmentRequest via his/her web browser, which forward 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 ARData is introduced to represent appointment requests in the group calendar. It has an assosiation with the class GroupCalendar named BelongsTo.
A new class DateData is introduced to represent dates in the appointment request. It has

an assosiation with the class ARData named BelongsTo.
A new class Participants is introduced to represent planned participants in the appointment request. It has an assosiation with the class ARData named BelongsTo.It has an assosiation with the class DateData named BelongsTo.

1.5.2 AnswerAR

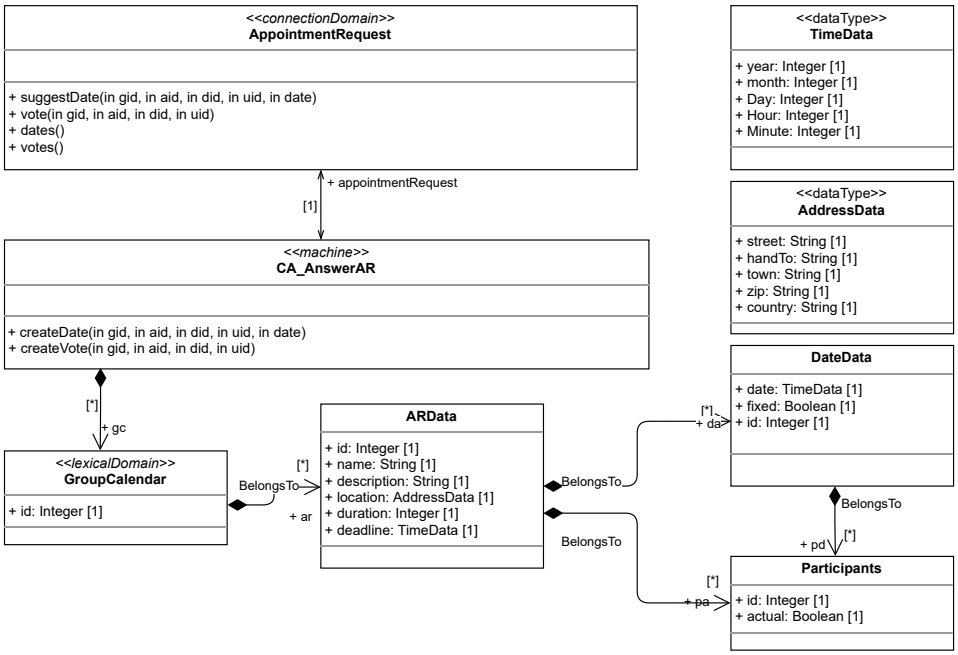

Figure 1.17: Class model: AnswerAR

Name: createDate
Description: Creates a new date for the appointment.
OCL constraint:

```
context CA_AnswerAR::createDate(gid:Integer, aid: Integer, uid:
    Integer, did: Integer, date: TimeData)
pre: gc->one(g: GroupCalendar | g.id=gid and g.ar->one(a: ARData |
     a.id=aid))
post: let g: GroupCalendar = gc->any(gr: GroupCalendar | gr.id=gid
    ) in
let a: ARData = g.ar->any(arq : ARData | arq.id = aid) in
a.da->one(d:DateData|d.id=did and d.date=date and d.fixed=false
    and d.pd->one(p Participants| p.id=uid)) and
AppointmentRequest^dates()
```

Name: suggestDate
Description: Forwards a new suggestion for a date from a group member to the machine.
OCL constraint:

```
context: AppointmentRequest::suggestDate(gid: Integer, aid:
    Integer, did: Integer, uid: Integer, date: TimeData)
pre: true
post: CA_AnswerAR^createDate(gid, aid, did, uid, date)
```

Group calendars have a unique id.
OCL constraint:

```
context: GroupCalendar
inv: GroupCalendar.allInstances()->isUnique(id)
```

Appointment requests have a unique id.
OCL constraint:

```
context: ARData
inv: ARData.allInstances()->isUnique(id)
```

Dates have a unique id.
OCL constraint:

```
context: DateData
inv: DateData.allInstances()->isUnique(id)
```

Participants have a unique id.
OCL constraint:

```
context: Participants
inv: Participants.allInstances()->isUnique(id)
```

Validation createDate

Operation specifications must be consistent with abstract specifications:
The operation specification of createDate 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 are covered in the postcondition.

Parameters must be used in the pre- and/or postcondition:
The parameters are used in the pre- and postcondition.

All parameters of operation must be known by the caller and all parameters of sent messages must be known by the machine:
GroupMember can input all parameters to AppointmentRequest via his/her web browser, which forward 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 ARData is introduced to represent appointment requests in the group calendar. It has an assosiation with the class GroupCalendar named BelongsTo.
A new class DateData is introduced to represent dates in the appointment request. It has an assosiation with the class ARData named BelongsTo.
A new class Participants is introduced to represent planned participants in the appointment request. It has an assosiation with the class ARData named BelongsTo.It has an assosiation with the class DateData named BelongsTo.

Name: createVote
Description: creates an entry for the vote
OCL constraint:

```
context CA_AnswerAR::createVote(gid: Integer, aid: Integer, did:
    Integer, uid: Integer)
pre: gc->one(g: GroupCalendar | g.id=gid and g.ar->one(a: ARData |
     a.id=aid))
post: let g: GroupCalendar = gc->any(gr: GroupCalendar | gr.id=gid
    ) in
let a: ARData = g.ar->any(arq : ARData | arq.id = aid) in
a.da->one(d:DateData|d.id=did and d.fixed=false and d.pd->one(p
    Participants| p.id=uid)) and
AppointmentRequest^votes()
```

Name: vote
Description: Forwards a vote for a date from a group member to the machine.
OCL constraint:

```
context: AppointmentRequest::vote(gid: Integer, aid: Integer, did:
     Integer, uid: Integer)
pre: true
post: CA_AnswerAR^createVote(gid, aid, did, uid)
```

Group calendars have a unique id.
OCL constraint:

```
context: GroupCalendar
inv: GroupCalendar.allInstances()->isUnique(id)
```

Appointment requests have a unique id.
OCL constraint:

```
context: ARData
inv: ARData.allInstances()->isUnique(id)
```

Dates have a unique id.
OCL constraint:

```
context: DateData
inv: DateData.allInstances()->isUnique(id)
```

Participants have a unique id.
OCL constraint:

```
context: Participants
inv: Participants.allInstances()->isUnique(id)
```

Validation createVote

Operation specifications must be consistent with abstract specifications:
The operation specification of vote 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 are covered in the postcondition.

Parameters must be used in the pre- and/or postcondition:
The parameters are used in the pre- and postcondition.

All parameters of operation must be known by the caller and all parameters of sent messages must be known by the machine:
GroupMember can input all parameters to AppointmentRequest via his/her web browser, which forward 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 ARData is introduced to represent appointment requests in the group calendar. It has an assosiation with the class GroupCalendar named BelongsTo.
A new class DateData is introduced to represent dates in the appointment request. It has an assosiation with the class ARData named BelongsTo.
A new class Participants is introduced to represent planned participants in the appointment request. It has an assosiation with the class ARData named BelongsTo.It has an assosiation with the class DateData named BelongsTo.

1.5.3 ShowCalendar

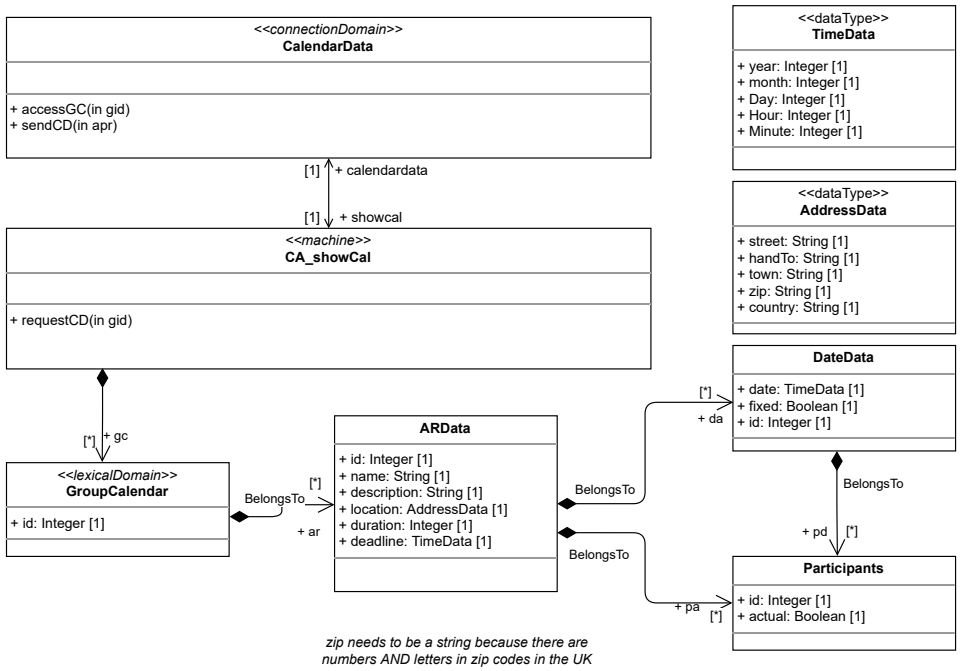

Figure 1.18: Class Model Show Calendar

Name: requestCD
Description: Requests the appointments and appointment requests of the calendar
OCL constraint:

```
context: CA_showCal :: requestCD(gid: Integer)
pre: gc->one(g: GroupCalendar | g.id=gid)
post: let apr: Set(ARData)=gc->any(g: GroupCalendar | g.id=gid and
    g.ar->asSet() in
Calendardata^sendCD(apr)
```

Name: accessGC
Description: Accesses the appointments and appointment requests of the calendar
OCL constraint:

```
context: CalendarData :: accessGC(gid: Integer)
pre: true
post: CA_showCal^requestCD(gid)
```

Group calendars have a unique id.
OCL constraint:

```
context: GroupCalendar
inv: GroupCalendar.allInstances()->isUnique(id)
```

Appointment requests have a unique id.
OCL constraint:

```
context: ARData
inv: ARData.allInstances()->isUnique(id)
```

Dates have a unique id.
OCL constraint:

```
context: DateData
inv: DateData.allInstances()->isUnique(id)
```

Participants have a unique id.
OCL constraint:

```
context: Participants
inv: Participants.allInstances()->isUnique(id)
```

Validation

Operation specifications must be consistent with abstract specifications:
The operation specification of requestCD 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 are covered in the postcondition.

Parameters must be used in the pre- and/or postcondition:
The parameters are used in the pre- and postcondition.

All parameters of operation must be known by the caller and all parameters of sent messages must be known by the machine:
GroupMember can input all parameters to CalendarData via his/her web browser, which forward these to this operation. The machine knows the argument arp used in the message to CalendarData.

All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
A new class ARData is introduced to represent appointment requests in the group calendar. It has an assosiation with the class GroupCalendar named BelongsTo.
A new class DateData is introduced to represent dates in the appointment request. It has an assosiation with the class ARData named BelongsTo.
A new class Participants is introduced to represent planned participants in the appointment request. It has an assosiation with the class ARData named BelongsTo.It has an assosiation with the class DateData named BelongsTo.

1.5.4 FinalizeAppointment

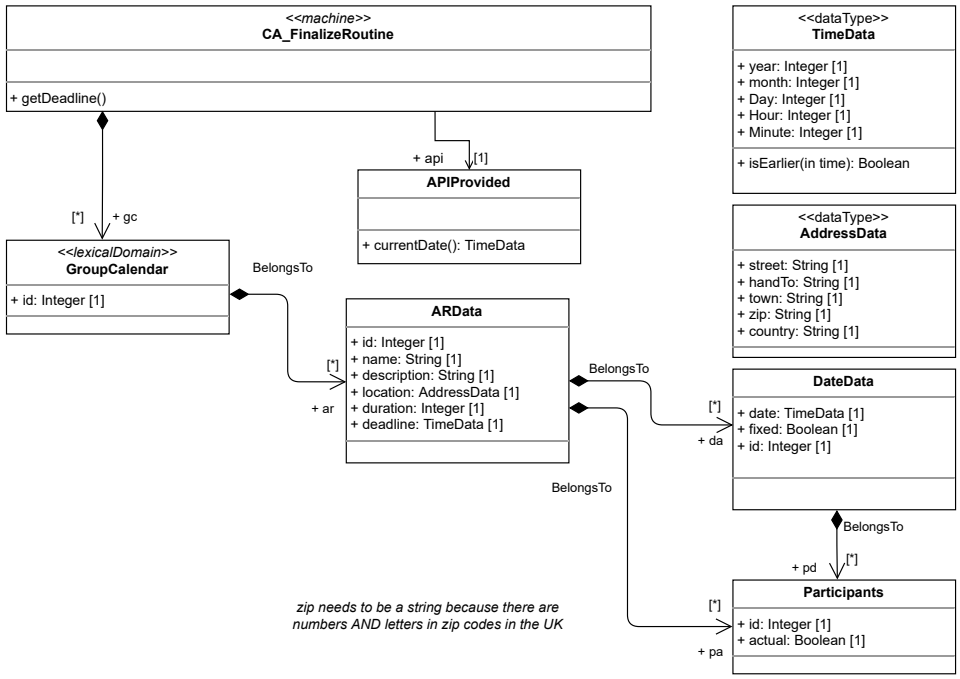

Figure 1.19: Class Model: FinalizeAppointment

Name: getDeadline
Description: This internal operation gets the deadline of an appointment request and checks if it is reached.
OCL constraint:

```
context: CA_FinalizeRoutine :: getDeadline()
pre: true
post: let g: GroupCalendar = gc->any(gr: GroupCalendar | gr.id=gid
    ) in
let appointments: Set(ARData)=g.ar@pre->select(a: ARData | a.
    deadline.isEarlier(api.currentDate()) and not(a.da->exists(d:
    DateData | d.fixed=true and d.pd->forAll(p: Participants | p.
    actual=true))) and a.da->exists(dt: DateData | dt.fixed=false
    and dt.pd->forAll(pt: Participants | pt.actual=false)))->asSet
    () in
appointments->forAll(a: ARData | a.da->one(d: DateData | d.fixed=
    true and d.pd->forAll(p: Participants | p.actual=true)) and not
    (a.da->exists(d: DateData | d.fixed=false)))
```

Group calendars have a unique id.
OCL constraint:

```
context: GroupCalendar
inv: GroupCalendar.allInstances()->isUnique(id)
```

Appointment requests have a unique id.
OCL constraint:

```
context: ARData
inv: ARData.allInstances()->isUnique(id)
```

Dates have a unique id.
OCL constraint:

```
context: DateData
inv: DateData.allInstances()->isUnique(id)
```

Participants have a unique id.
OCL constraint:

```
context: Participants
inv: Participants.allInstances()->isUnique(id)
```

Validation

Operation specifications must be consistent with abstract specifications:
The operation specification of getDeadline 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 are covered in the postcondition.

Parameters must be used in the pre- and/or postcondition:
The parameters are used in the postcondition.

All parameters of operation must be known by the caller and all parameters of sent messages must be known by the machine:
The operation does not contain parameters.

All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
A new class ARData is introduced to represent appointment requests in the group calendar. It has an assosiation with the class GroupCalendar named BelongsTo.
A new class DateData is introduced to represent dates in the appointment request. It has an assosiation with the class ARData named BelongsTo.
A new class Participants is introduced to represent planned participants in the appointment request. It has an assosiation with the class ARData named BelongsTo.It has an assosiation with the class DateData named BelongsTo.

1.6 A6

$LC_{GroupMember} = (showCalendar|addAppointment|AnswerAR)^*$
$LC_{CalendarApp} = (||_{i=1}^{n} LC_{GroupMember_i})||FinalizeAppointment$

1.6.1 Validation I

Each sequence diagram of A3 is contained in at least one life-cycle expression:

Table 1.23: A6 Validation I

scenario	life-cycle expression
sdAddAppointment	$LC_{GroupMember}$
sdAnswerAR	$LC_{GroupMember}$
sdShowCalendar	$LC_{GroupMember}$
sdFinalizeAppointment	$LC_{CalendarApp}$

1.6.2 Validation II

For each biddable domain exists exactly one life-cycle:
For the biddable domain GroupMember exactly one life-cycle exists, namely $LC_{GroupMember}$.

1.6.3 Validation III

The Life-Cycles are consistent with the state predicates in A3:
ShowCalendar has no state predicates at the beginning and end. Hence, it can be executed an arbitrary number of times.
AddAppointment has no state predicate at the beginning. Hence, it can be executed an arbitrary number of times.
AnswerAR can be executed if a AppointmentRequest object is created beforehand. This can be ensured by executing AddAppointment before executing AnswerAR. Otherwise, AnswerAR returns an empty set and no AppointmentRequest can be selected to vote or to suggest a new date on.
FinalizeAppointment has no state predicates at the beginning. Hence, it can be executed an arbitrary number of times.

1.6.4 Validation IV

The Life-Cycles are consisten with the pre- and post conditions of A5:
The sequence diagram AddAppointment contains the operation makeAR. It has only the precondition that the Group Calendar exists, so it has no precondition for the group member. Hence, it can be executed at any position of the life-cycle of the group member.
The sequence diagram AnswerAR contains the operations createDate and createVote. createVote and createDate requires, that the calendar exists and an appointment request with the supplied aid exists. This is ensured by the postcondition of makeAR, that creates an AppointmentRequest with an aid. Only the aid, that ist already created by makeAR can be an input for createVote and createDate. Hence, AddAppointment must be xecuted before AnswerAR.
The sequence diagram showCalendar contains the operation requestCD. It has only the precondition that the calendar exists, so it has no precondition for the group member. Hence,

it can be executed at any postion of the life-cycle.
The sequence diagram FinalizeAppointment contains the operation getDeadline. It has no precondition. Hence, it can be executed at any position of the life-cycle.

1.6.5 Validation V

Exactly one life-cycle exists for the machine domain, that combines all life-cycles:
The life-cycle for the machine domain is $LC_{Calendar\_App}$. It combines all life-cycles.

2 Design

2.1 D1

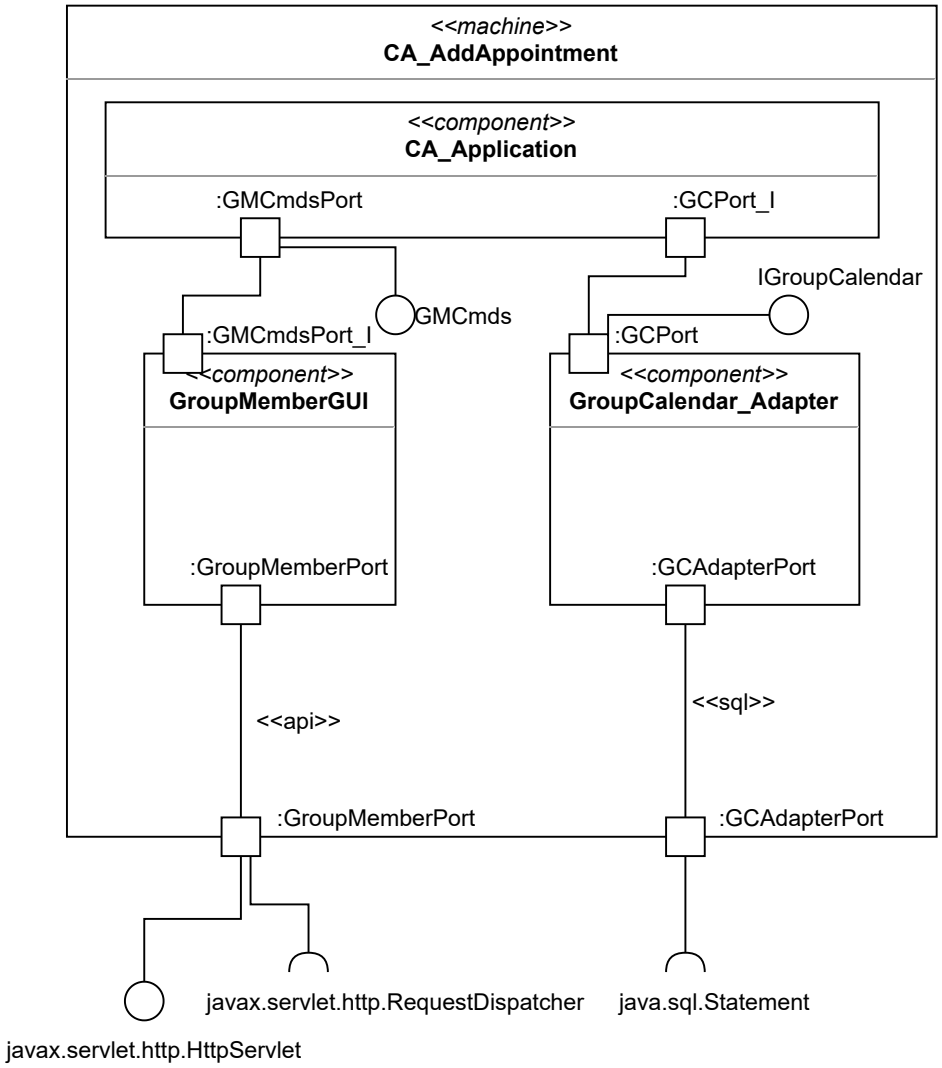

Figure 2.1: Architectural Pattern for CA_AddAppointment(subArchAddAppointment)

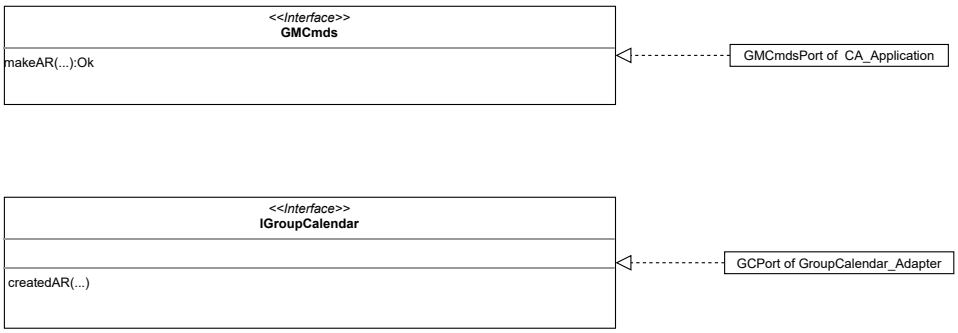

Figure 2.2: Internal Interfaces AddAppointment

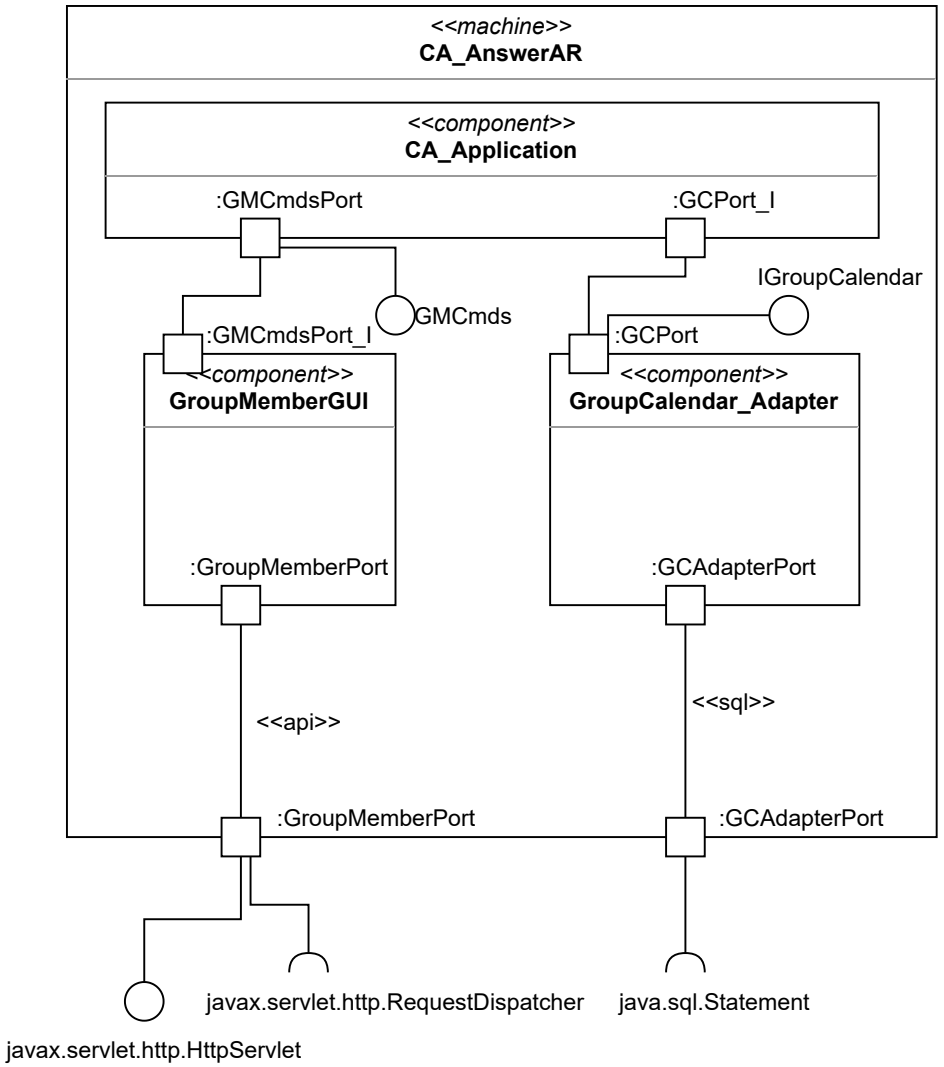

Figure 2.3: Architectural Pattern for CA_AnswerAR (subArchAnswerAR)

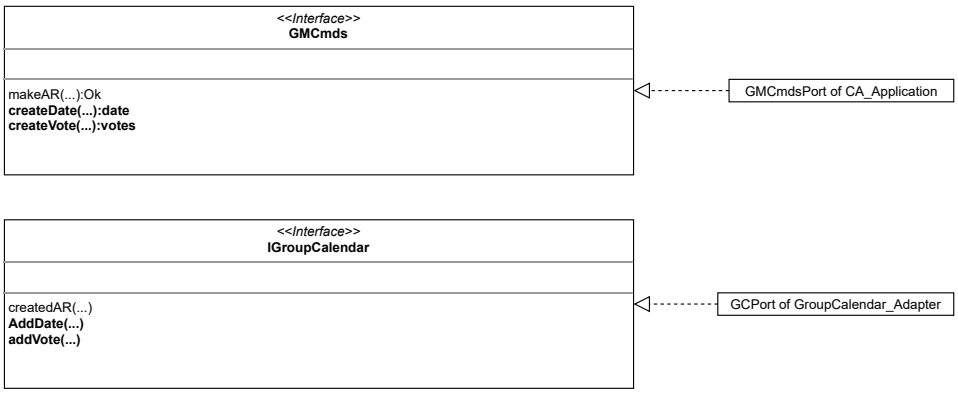

Figure 2.4: Internal Interfaces AnswerAR

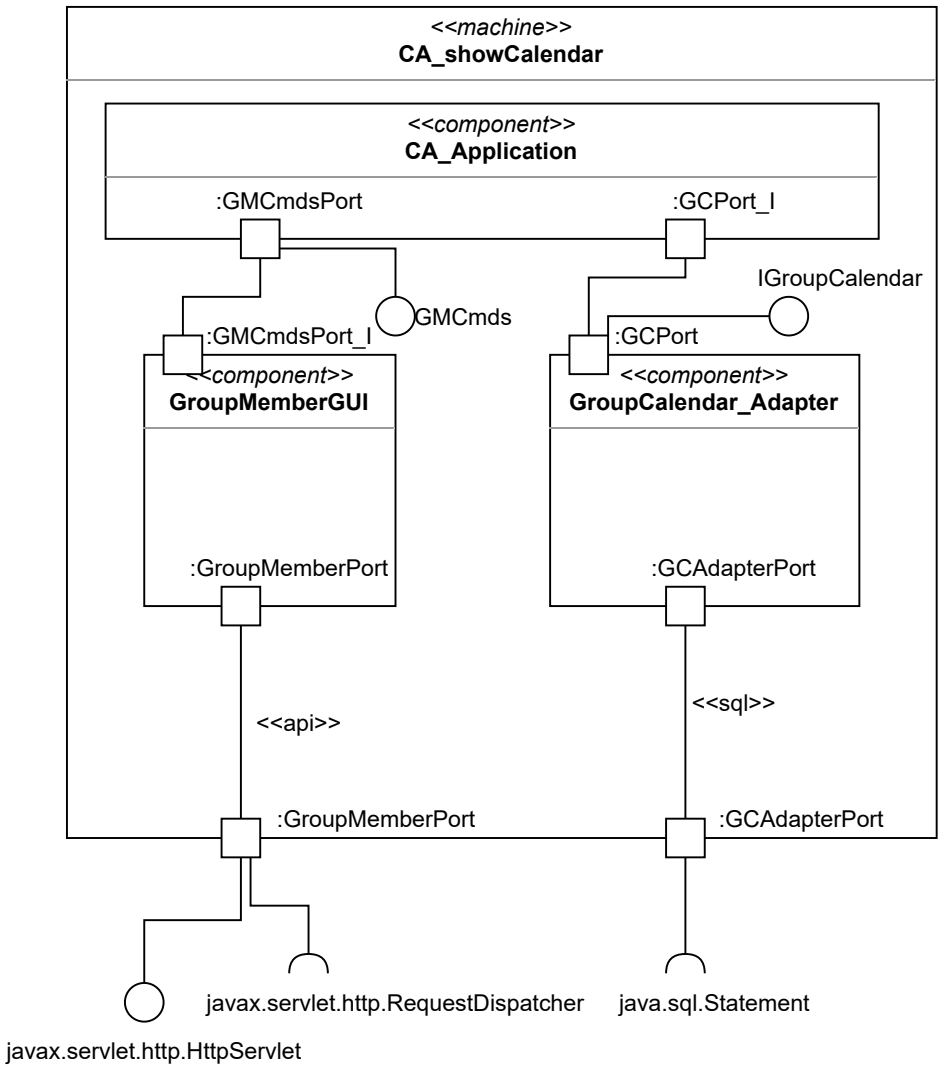

Figure 2.5: Architectural Pattern for CA_showCalendar (subArchShowCalendar)

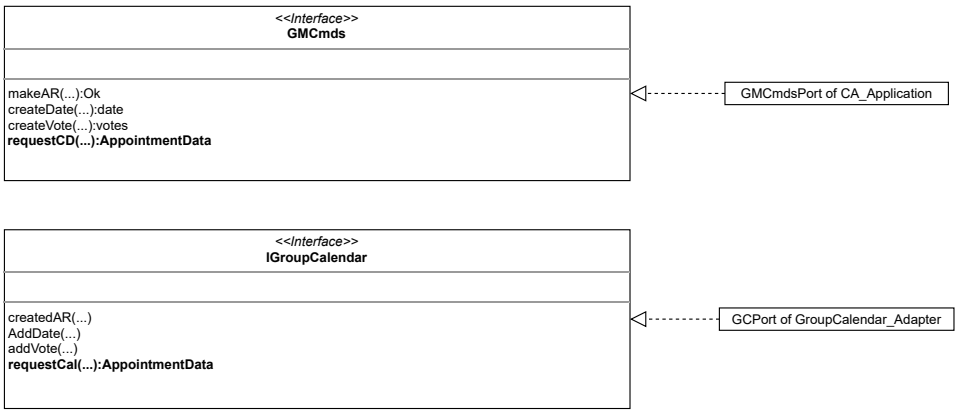

Figure 2.6: Internal Interfaces ShowCalendar

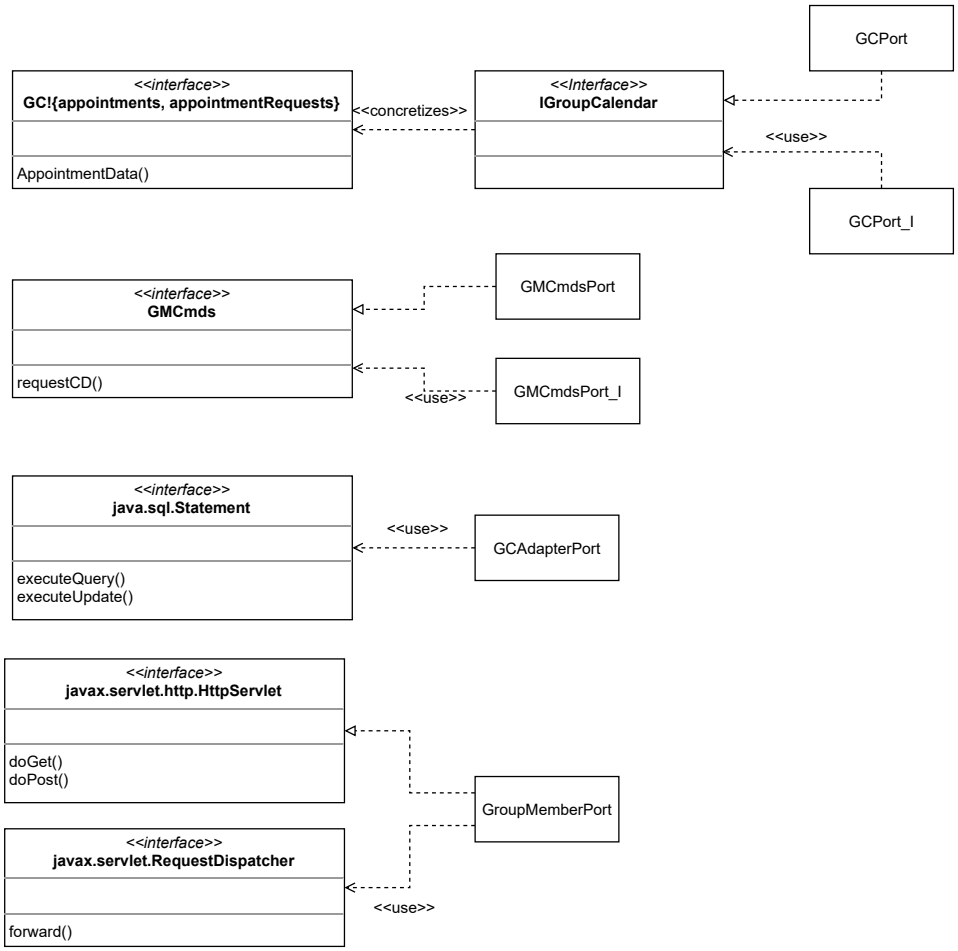

Figure 2.7: Port types and interface relations for ShowCalendar

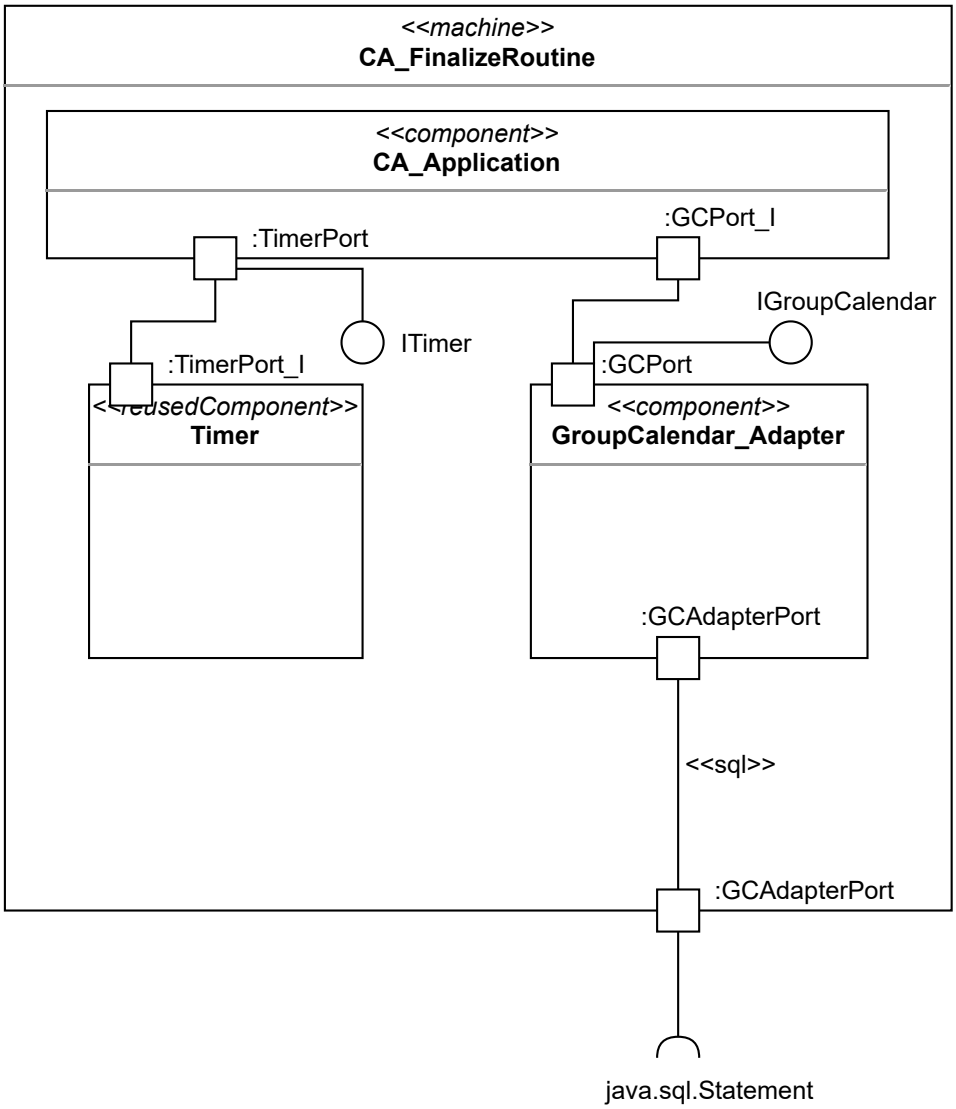

Figure 2.8: Architectural Pattern for CA_FinalizeRoutine (subArchFinalizeRoutine)

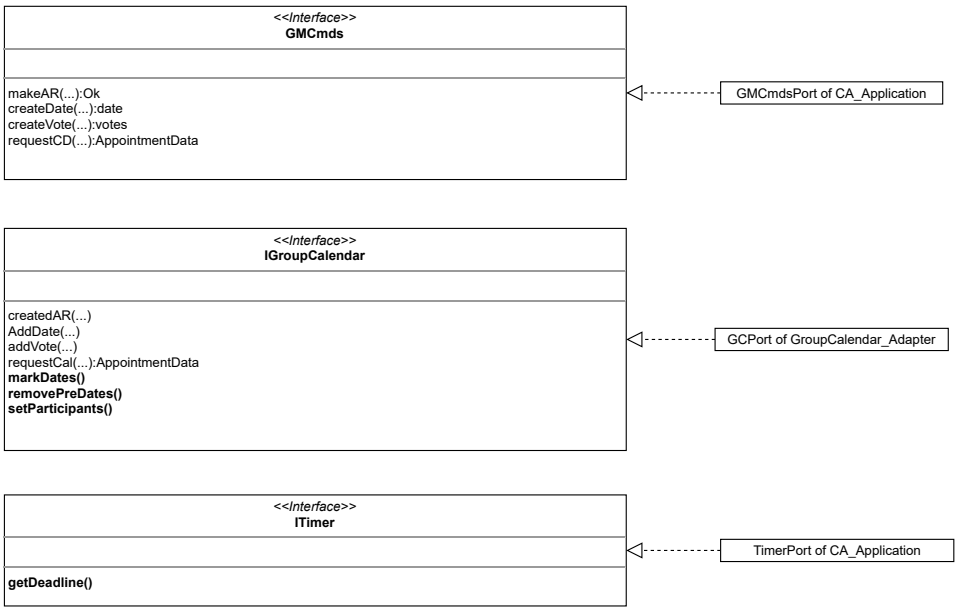

Figure 2.9: Internal Interfaces FinalizeRoutine

Refining app_if interface classes:

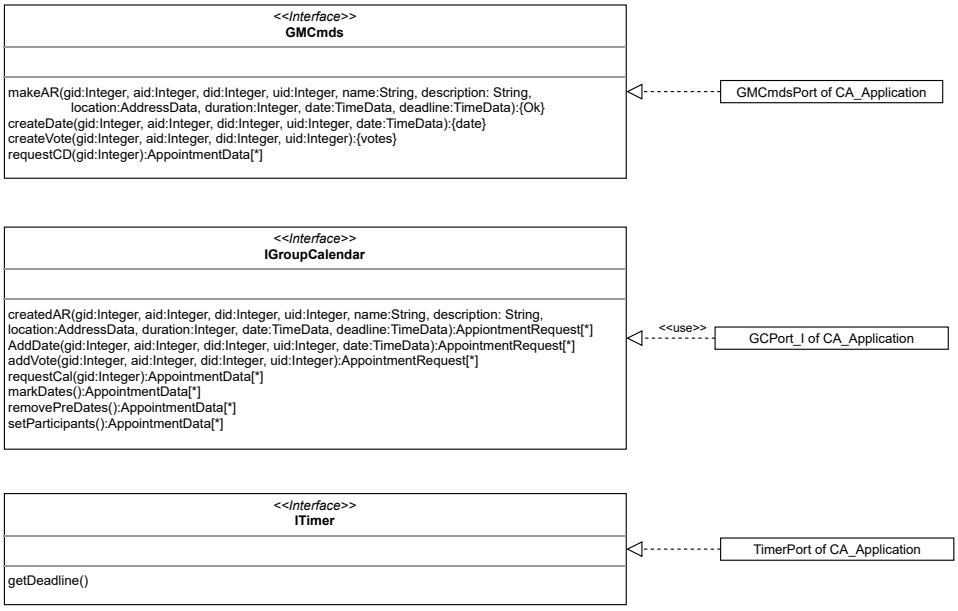

Figure 2.10: Refining appif

Refining tech_if" interface classes:

Table 2.1: Refining tech_if" interface classes

considered interface in subproblem architecture	technical interface
<< api >> javax.servlet.http.HttpServlet in CA_AddAppointment	<< api >> AT!{doGet,doPost}
<< api >> javax.servlet.RequestDispatcher in CA_AddAppointment	<< api >> CA!{forward}

<< sql >> java.sql.Statement in CA_AddAppointment	<< call_return, sql >> CA!{executeQuery, executeUpdate}
<< api >> javax.servlet.http.HttpServlet in CA_AnswerAR	<< api >> AT!{doGet,doPost}
<< api >> javax.servlet.RequestDispatcher in CA_AnswerAR	<< api >> CA!{forward}
<< sql >> java.sql.Statement in CA_AnswerAR	<< call_return, sql >> CA!{executeQuery, executeUpdate}
<< api >> javax.servlet.http.HttpServlet in CA_showCalendar	<< api >> AT!{doGet,doPost}
<< api >> javax.servlet.RequestDispatcher in CA_showCalendar	<< api >> CA!{forward}
<< sql >> java.sql.Statement in CA_showCalendar	<< call_return, sql >> CA!{executeQuery, executeUpdate}
<< sql >> java.sql.Statement in CA_FinalizeRoutine	<< call_return, sql >> CA!{executeQuery, executeUpdate}

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.

Merging subproblem architecture:
The components can be merged as follows:
The application components in all architectures for the subproblem should be merged because the Subproblems AddAppointment, AnswerAR and showCalendar are realted sequentially. The subproblem FinalizeRoutine is also merged because of reasons of simplicity.
The GroupCalendar_Adapters in all architectures for the subproblems should be merged because it is an adapter, establishing the connection to the DB.

The Timer is used only in one subproblem.
The GroupMemberGUI for the Group Member in all architectures for the subproblems uses the same technology and should be merged.

Reusable Components:
The Timer should trigger the FinalizeRoutine sequence.

Components to Develop:
We have to develop the CA_Application, a GroupMemberGUI and the adapter for the external components.
The GroupCalendar_Adapter is respononsible to create and maintain table for all persistent classes.

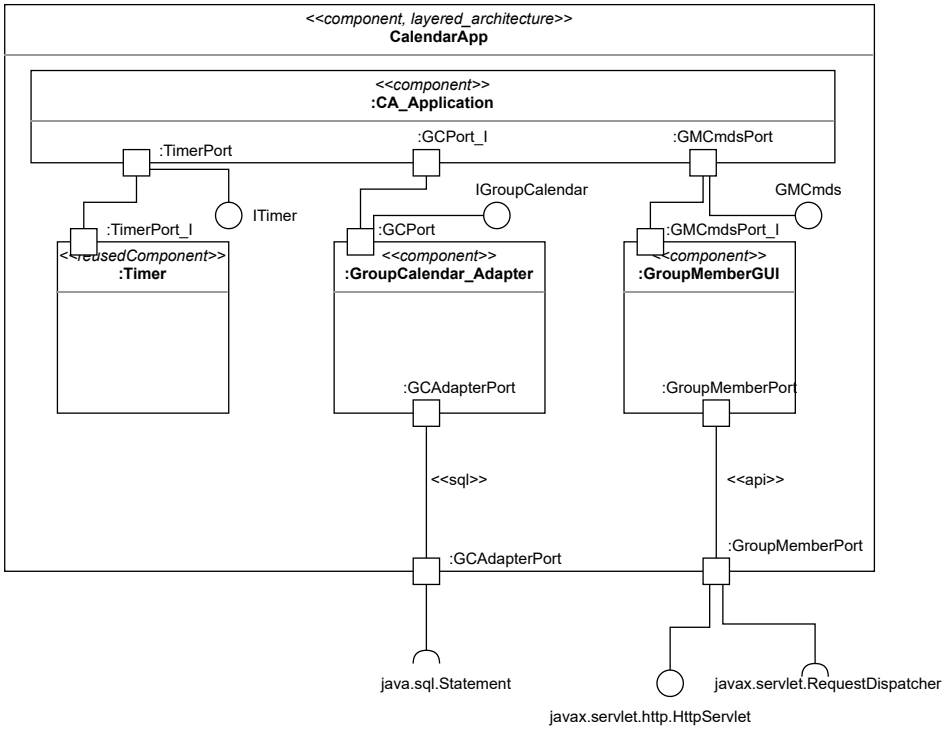

Figure 2.11: Merged Architecture for CalendarApp (globalArch)

2.1.1 Validation

Validation I

Table 2.2: D1 Validation I

squence diagram	message	in/out	Application Layer Interface	required/ provided
AddAppointment	makeAR	in	GMCmdsPort::makeAR	provided
	createdAR	out	GCPort::createdAR	required
	AppointmentRequest	in	return value of GCPort::createdAR	required
	ShowOk	out	return value of GMCmdsPort::makeAR	provided
AnswerAR	createDate	in	GMCmdsPort::createDAte	provided
	AddDate	out	GCPort::AddDate	required
	AppointmentRequest	in	return value of GCPort::AddDate	required
	dates	out	return value of GMCmdsPort::createDAte	provided
	createVote	in	GMCmdsPort::createVote	provided
	addVote	out	GCPort::addVote	required
	AppointmentRequest	in	return value of GCPort::addVote	required
	votes	out	return value of GMCmdsPort::createVote	provided
showCalendar	requestCD	in	GMCmdsPort::requestCD	provided
	requestCal	out	GCPort::requestCal	required
	AppointmentData	in	return value of GCPort::requestCal	required
	sendCD	out	return value of GMCmdsPort::requestCD	provided
FinalizeAppointment	getDeadline	in	ITimer::getDeadline	provided
	markDates	out	GCPort::markDates	required
	removePreDates	out	GCPort::removePreDates	required
	setParticipants	out	GCPort::setParticipants	required

Validation II

For global architecture: direction of all messages consitent to each other and input

Table 2.3: D1 Validation II.I

provided by machine	required by adapter/provided by app
javax.servlet.http.HttpServlet	GMCmds
-	ITimer

Table 2.4: D1 Validation II.II

required by machine	provided by adapter/required by app
javax.servlet.RequestDispatcher	return values in GMCmds
java.sql.Statement	IGroupCalendar

Validation III

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

Table 2.5: D1 Validation III

external port type	interface in architecture	required/pro	interface in technical context diagram
GroupMemberPort	javax.servlet.http.HttpServlet	provided	AT!{doGet,doPost}
	javax.servlet.RequestDispatcher	required	VR!{forward}
GCAdapterPort	java.sql.Statement	required	CA!{executeQuery, executeUpdate}

2.2 D2

2.2.1 AddAppointment

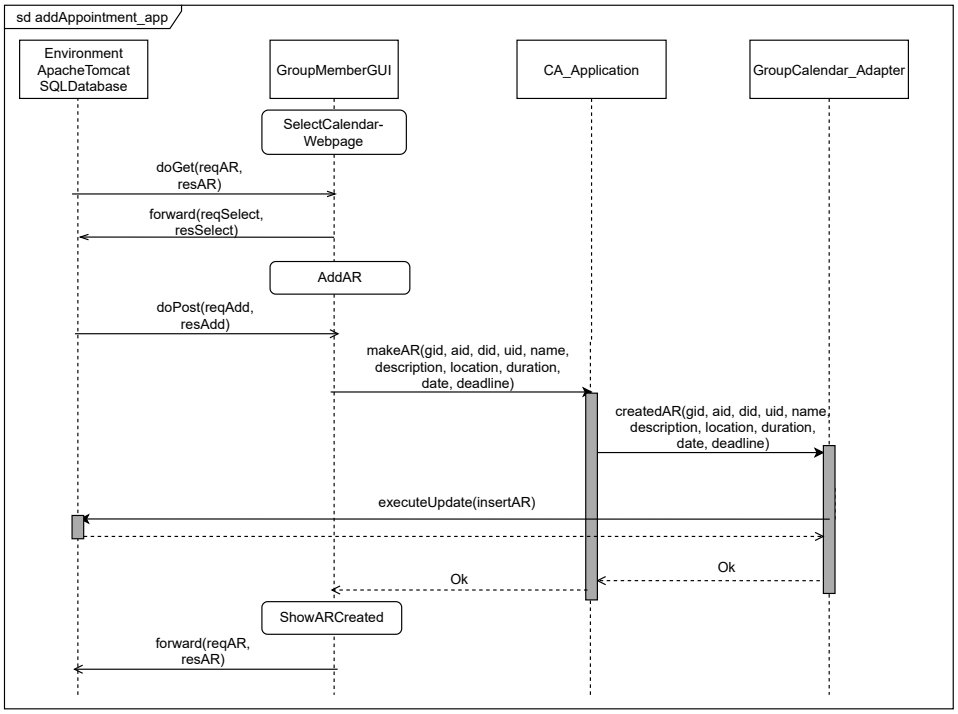

Figure 2.12: sdAddAppointment_app

Remarks:
reqSelect and reqAdd represent HTTPServletRequest objects containing teh required user input.
resSelect and resAdd represent HttpServletResponse objects as the counterpart for the request.
The state predicate SelectCalendarWebpage represents that the requested Calendar is shown.
The state predicate AddAR represents that the input form for creating an appointmentRequest is shown.
The state predicate ShowARCreated represents that the confirmation of the creation of the Appointment is shown.

forward(...) sends the request and response back to the server to generate the HTML webpage.

Since we use a MySQL database, we do not need to specify the interfaces to lexical doamins in more detail. We use standadized SQL statements to acces the database.

insertAR:

```
INSERT INTO appointments
            (gid,
             aid,
             name,
             description,
             locationHandTo,
             locationStreet,
             locationTown,
             locationZIP,
             locationCountry,
             duration,
             deadline)
VALUES      ("gid",
             "aid",
             "name",
             "description",
             "locationHandTo",
             "locationStreet",
             "locationTown",
             "locationZIP",
             "locationCountry",
             "duration",
             "deadline");

INSERT INTO dates
            (aid,
             did,
             date,
             fixed)
VALUES      ("aid",
             "did",
             "date",
             false);

INSERT INTO participants
            (aid,
             uid,
             actual)
VALUES      ("aid",
             "uid",
             false);
```

2.2.2 AnswerAR

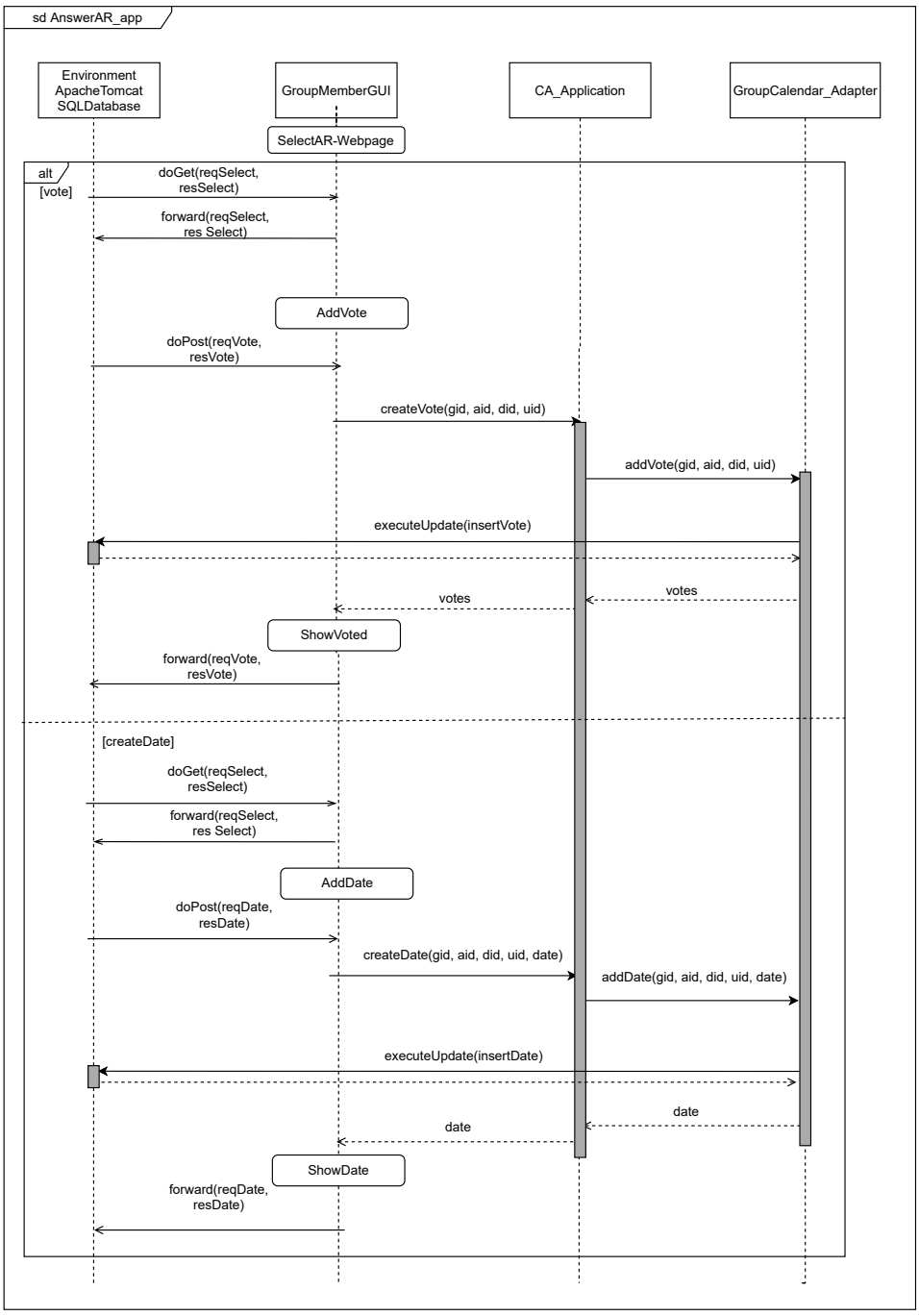

Figure 2.13: sdAnswerAR_app

Remarks:
reqSelect,reqVote & reqDate represents HttpServletRequest objects containing the required user input.
resSelect, resVote & resDate represents HttpServletResponse objects as the counterpart for the request.
The state predicate SelectAR-Webpage represents that the requested appointmentRequest is shown.

The state predicate SelectCalendarWebpage represents that the requested Calendar is shown.
The state predicate AddVote represents that the input form for voting the selected appointmentRequest is shown.
The state predicate AddDate represents that the input form for creating a date the selected appointmentRequest is shown.
The state predicate ShowVoted represents that the confirmation of the voting is shown.
The state predicate ShowDate represents that the confirmation of the addition of a new date is shown.
forward(...) sends the request and response back to the server to generate the HTML webpage.

Since we use a MySQL database, we do not need to specify the interfaces to lexical domains in more detail. We use standardized SQL statements to access the database.

insertVote:

```
1  INSERT INTO dates2participants
2              (did,
3               uid)
4  VALUES      ("did",
5               "uid");
```

insertDate:

```
1  INSERT INTO dates
2              (aid,
3               did,
4               date,
5               fixed)
6  VALUES      ("aid",
7               "did",
8               "date",
9               false)
10
11 INSERT INTO dates2participants
12             (did,
13              uid)
14 VALUES      ("did",
15              "uid");
```

2.2.3 ShowCalendar

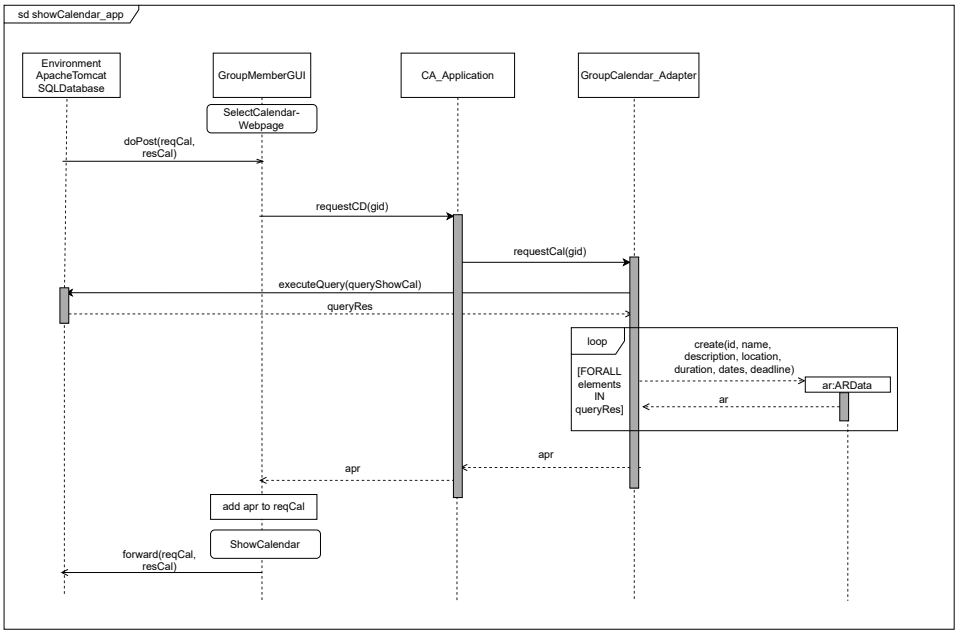

Figure 2.14: sdShowCalendar_app

Remarks:
reqCal represents a HttpServletRequest object containing the required user input.
resCal represents a HttpServletResponse object as the counterpart for the request.
forward(...) sends the request and response back to the server to generate the HTML webpage.
The state predicate ShowCalendar represents that the dates of the selected group calendar is shown.

Since we use a MySQL database, we do not need to specify the interfaces to lexical domains in more detail. We use standardized SQL statements to access the database.

queryShowCal:

```
1  SELECT a.aid,
2         d.did,
3         a.name,
4         a.description,
5         a.locationHandTo,
6         a.locationStreet,
7         a.locationTown,
8         a.locationZIP,
9         a.locationCountry,
10        a.duration,
11        d.date,
12        a.deadline
13 FROM    appointments AS a
14 join dates AS d on a.aid = d.aid
15 WHERE  a.gid = "gid";
```

2.2.4 FinalizeAppointment

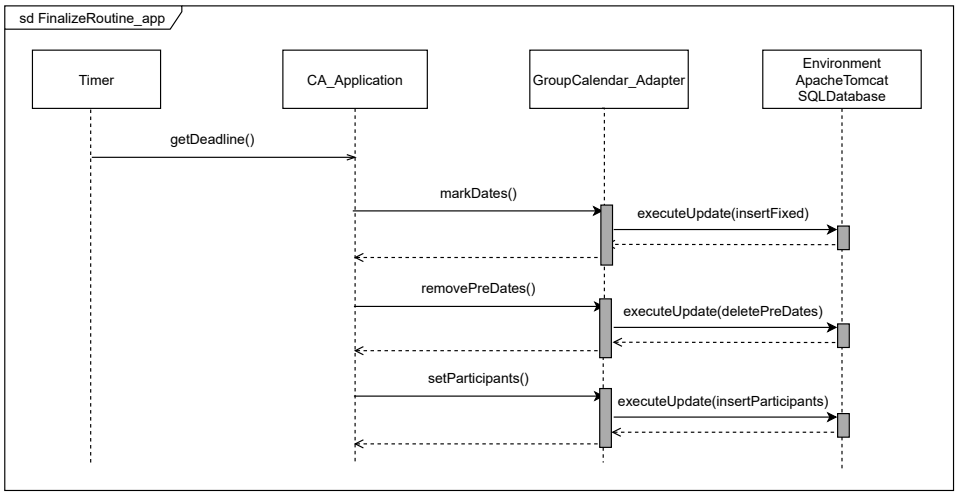

Figure 2.15: sdFinalizeRoutine_app

Since we use a MySQL database, we do not need to specify the interfaces to lexical domains in more detail. We use standardized SQL statements to access the database.

insertFixed:

```
UPDATE dates
SET    fixed = 1
WHERE  did IN (SELECT did
               FROM   dates
               WHERE  did IN (SELECT p1.did
                              FROM   (SELECT did,
                                             Count (uid) AS psum
                                      FROM   dates2participants
                                      GROUP  BY did) AS p1,
                                     (SELECT p2.aid,
                                             Max(p2.psum) AS psum
                                      FROM   (SELECT ad2p.aid,
                                                     Count(ad2p.uid) AS psum
                                              FROM
                                             (SELECT a.aid,
                                                     d.did,
                                                     d2p.uid
                                              FROM   appointments AS a
                                                     JOIN dates AS d
                                                       ON a.aid = d.aid
                                                     JOIN dates2participants AS
                                                          d2p
                                                       ON d.did = d2p.did
                                              WHERE  CURRENT_TIMESTAMP >
                                              a.deadline)
                                             AS ad2p
                                              GROUP  BY ad2p.did,
                                                        ad2p.aid) AS p2
                                      GROUP  BY p2.aid) AS p3
                              WHERE  p1.psum = p3.psum
                                     AND dates.aid = p3.aid
                                     AND dates.did = p1.did));
```

deletePreDates:

```
DELETE FROM dates
WHERE  fixed = 0
       AND aid IN (SELECT a.aid
                   FROM   appointments AS a
                   WHERE  CURRENT_TIMESTAMP > a.deadline);
```

insertParticipants:

```
UPDATE participants
SET    actual = 1
WHERE  uid IN (SELECT ad2p.uid
               FROM   (SELECT d2p.did,
                              d2p.uid
                       FROM   appointments AS a
                              JOIN dates AS d
                                ON a.aid = d.aid
                              JOIN dates2participants AS d2p
                                ON d.did = d2p.did
                       WHERE  CURRENT_TIMESTAMP > a.deadline) AS ad2p
               WHERE  ad2p.did = (SELECT did
                                  FROM   dates
                                  WHERE  fixed = 1
                                         AND aid = participants.aid));
```

2.2.5 Validation

Validation I

The sequence diagram must be consistent with the behavior decribed in Step A3 and in Step A6:

Table 2.6: Consistency of sdAddAppointment_app and sdAddAppointment

Message in D2	Corresponding Message in A3
doGet(reqSelect,resSelect)	refines createAR
forward(reqSelect,resSelect)	refines createAR
doPost(reqAdd,resAdd)	refines createAR
makeAR(...)	makeAR
createdAR(...)	createdAR
executeUpdate(insertAR)	refines createdAR
forward(reqAdd,resAdd)	refines okRepresentation

Table 2.7: Consistency of sdAnswerAR_app and sdAnswerAR

Message in D2	Corresponding Message in A3
doGet(reqSelect,resSelect)	refines suggestDate, vote
forward(reqSelect,res Select)	refines suggestDate, vote
doPost(reqVote,resVote)	refines vote
createVote(...)	createVote
addVote(...)	addVote
executeUpdate(insertVote)	refines addVote
forward(reqVote,resVote)	refines showVotes
doPost(reqDate,resDate)	refines suggestDate
createDate(...)	createDate
addDate(...)	addDate
executeUpdate(insertDate)	refines addDate
forward(reqDate,resDate)	refines showDates

Table 2.8: Consistency of sdShowCalendar_app and sdShowCalendar

Message in D2	Corresponding Message in A3
doGet(reqCal,resCal)	refines accessGC
requestCD(...)	requestCD
requestCal(...)	requestCal
executeQuery(queryShowCal)	refines requestCal
forward(reqCal,resCal)	refines sendCalendarData

Table 2.9: Consistency of sdFinalizeRoutine_app, sdFinalizeRoutine

Message in D2	Corresponding Message in A3
getDeadline()	getDeadline
markDates()	markDates
executeUpdate(insertFixed)	refines markDates
removePreDates()	removePreDates
executeUpdate(deletePreDates)	refines removePreDates
setParticipants()	setParticipants
executeUpdate(insertParticipants)	refines setParticipants

Consistency with Lifecycles
$(showCalendar|addAppointment|AnswerAR)^*$:
sdshowCalendar_app, sdAddAppointment_app and sdAnswerAR_app can be executed an arbitrary times without precondition.
$(showCalendar|addAppointment|AnswerAR)^*||FinalizeAppointment$:
The sequence of sdShowCalendar_app, sdaddAppointment_app and sdAnswerAR_app can be executed concurrently with sdFinalizeRoutine_app without unwanted side-effects.

Validation II

The sequence diagrams must realize the operations decribed in Step A5:
makeAR(...) is realized in sdAddAppointment_app
Precondition is established, because before makeAR ist executed, an existing group calendar is selected by the message doGet(reqSelect,resSelect).
Postcondition is established, because using the SQL command insertAR, the appointment request is added. The added appointment request is returned to CA_Application. With Ok the GroupMemberGUI is instructed to show the okRepresentation.

createDate(...) is realized in sdAnswerAR_app
Precondition is established, because before createDate is executed, an existing group calendar and an existing appointment request is selected by the message doGet(reqSelect,resSelect).
Postcondition is established, because useing the SQL command insertVote a new vote is added to the date of the selected appointment request. The updatet apppointment request is returned to CA_Application. By the return value votes the GroupMemberGUI is instructed to show the votes for the appointment request.

createVote(...) is realized in sdAnswer_app
Precondition is established, because before createDate is executed, an existing group calendar and an existing appointment request is selected by the message doGet(reqSelect,resSelect).
Postcondition is established, because using the SQL command insertDate a new date is added to the appointment request. The updatet appointment request is returned to CA_Application. By the return value dates the GroupMemberGUI is instructed to show the dates for the appointment request.

requestCD(...) is realized in sdShowCalendar_app
Precondition is established, because before requestCD is executed an existing group calendar is selected by the message selectGroupCalendar(gid).
Postcondition CA_Application delegates the message to GroupCalendar_Adapter. Using the SQL command gueryShowCal, GroupCAlendar_Adapter selects all appointment requests and all appointments for the selected calendar. All this informations are then returned to CA_Application. CA_Application forwards the result to GroupMemberGUI. That realizes calendardatasendCD(apr)

getDeadline() is realized in sdFinalizeRoutine_app
Precondition does not have to be established, because it is true.
Postcondition is established because insertFixed marks the date that is now chosen for the appointment as fixed. The SQL command deletePreDates removes all other dates, that are not fixed and insertParticipants sets all planned Participants that can participate at the fixed date as actual participants

Validation III

All messages in the application interface classes of step D1 must be used in some sequence diagram.

Table 2.10: D2 Validation III

Interface	Message	Used in sequence diagram
GMCmds	makeAR	sdAddAppointment_app
	createDate	sdAnswerAR_app
	createVote	sdAnswerAR_app
	requestCD	sdShowCalendar_app
IGroupCalendar	createAR	sdAddAppointment_app
	addDate	sdAnswerAR_app
	addVote	sdAnswerAR_app
	requestCal	sdShowCalendar_app
	markDates	sdFinalizeRoutine_app
	removePreDates	sdFinalizeRoutine_app
	setParticipants	sdFinalizeRoutine_app
ITimer	getDeadline	sdFinalizeRoutine_app

Validation IV

Table 2.11: D2 Validation IV

Interface	Provided by	Required by
Message	Recipient	Sender
GMCmds	CA_Application	GroupMemberGUI
makeAR	CA_Application	GroupMemberGUI
createDate	CA_Application	GroupMemberGUI
createVote	CA_Application	GroupMemberGUI
requestCD	CA_Application	GroupMemberGUI
IGroupCalendar	GroupCalendar_Adapter	CA_Application
createAR	GroupCalendar_Adapter	CA_Application
addDate	GroupCalendar_Adapter	CA_Application
addVote	GroupCalendar_Adapter	CA_Application
requestCal	GroupCalendar_Adapter	CA_Application
markDates	GroupCalendar_Adapter	CA_Application
removePreDates	GroupCalendar_Adapter	CA_Application
setParticipants	GroupCalendar_Adapter	CA_Application
ITimer	CA_Application	Timer
getDeadline	CA_Application	Timer

Validation V

Messages must connect components as connected in the software architecture of step D1:

Table 2.12: D2 Validation V

Component	Connected components in architecture	Connected components in sequence diagrams
CA_Application	GroupMemberGUI, GroupCalendar_Adapter, Timer	GroupMemberGUI, GroupCalendar_Adapter, Timer
GroupCalendar_Adapter	CA_Application, Environment	CA_Application, Environment
GroupMemberGUI	CA_Application, Environment	CA_Application, Environment
Timer	CA_Application	CA_Application

Hence, for all components the sets of connected components in the architecture and in the sequence diagrams are the same.

2.3 D3

We do not need to do this step, because our components are not complex enough for this step to be required.
So we do not have to split our components in sub-components.
AddAppointment:
The only component that could be to complex is the GroupCalendar_Adapter. But this component is also not so complex because it only sends a SQL query to the database management system and forwards the confirmation of the query to the CA_Application.
Answer_AR:
The only component that could be to complex is the GroupCalendar_Adapter. But this component also is not so complex because it only sends a SQL query to the database management system and forwards the confirmation of the query to the CA_Application.
ShowCalendar:
The only component that could be to complex is the GroupCalendar_Adapter. But this component also is not so complex because it only sends a SQL query to the database management system and creates an object of the class ARData for every result of the query and forwards this objects to the CA_Application.
FinalizeAppointment:
The only component that could be to complex is the GroupCalendar_Adapter. But this component also is not so complex because it only sends a SQL query to the database management system.

2.4 D4

Check whether a state machine is necessary:
The component Timer is re-used components. It is not necessary to create state machines for it.
The component CA_Application: There is no refinement of this component in Step D3; continue looking at Step D2. Most of the time, the machine gets an input message that is passed on. The machine then waits for the results. Furthermore, the life-cycle is ensured via the group member. It is not necessary to create a state machine for this component.
The component CA_Adapter: There is no refinement of this component in Step D3; continue looking at Step D2. It is not necessary to create a state machine for this component, because the data base with its corresponding DBMS handles the states and the state changes.
The component GroupMemberGUI: No refinement exists in Step D3; continue with looking at Step D2. There are more than two states. Therefore, a state machine is required.

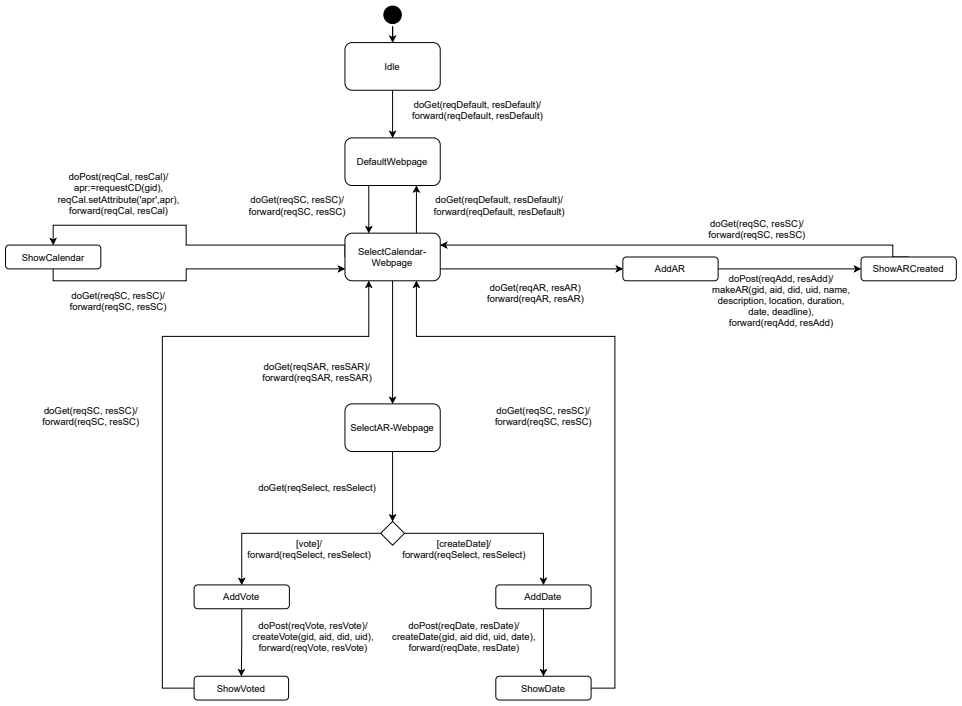

Figure 2.16: State machine GroupMemberGUI

Remarks:
We add the following additional transitions to model the complete behaviour of the web application:

- *doGet(reqDefault, resDefault)* is a trigger that represents the initial request for the webpage when entering the URL.
- *forward(reqDefault, resDefault)* is the corresponding action to generate the starting page.
- *setAttribute('name', value)* is a method to use results in a generated webpage.
- *doGet(reqSC, resSC)* is a trigger that represents the request for the Overview page of a group calendar.
- *forward(reqSC, resSC)* is the corresponding action to generate the Overview page.
- *doGet(reqSAR, resSAR)* is a trigger that represents the request for the Overview page of an appiontment request.
- *forward(reqSAR, resSAR)* is the corresponding action to generate the Overview page.

2.4.1 Validation

Validation I

The state machines describe the same behavior as in Step D2 or D3:

Table 2.13: Consistency of state machine and D2 and D3

Component GroupMemberGUI					
Source State	**Target State**	**Input Signal**	**Mapped to Message(s)**	**Output Signal**	**Mapped to Message(s)**
Init	DefaultWebpage	doGet (reqDefault, resDefault)	-	forward (reqDefault, resDefault)	-
DefaultWebpage	SelectCalendar-Webpage	doGet (reqPage, resPage)	-	forward (reqPage, resPage)	-
SelectCalendar-Webpage	SelectAR-Webpage	doGet (reqPage, resPage)	-	forward (reqPage, resPage)	-
SelectCalendar-Webpage	DefaultWebpage	doGet (reqDefault, resDefault)	-	forward (reqDefault, resDefault)	-
SelectCalendar-Webpage	ShowCalendar	doPost (reqCal, resCal)	doPost (reqCal, resCal)	forward (reqCal, resCal)	forward(reqCal, resCal)
ShowCalendar	SelectCalendar-Webpage	doGet (reqPage, resPage)	-	forward (reqPage, resPage)	-
SelectCalendar-Webpage	AddAR	doGet (reqSelect, resSelect)	doGet (reqSelect, resSelect)	forward (reqSelect, resSelect)	forward (reqSelect, resSelect)
AddAR	ShowARCreated	doPost (reqAdd, resAdd)	doPost (reqAdd, resAdd)	forward (reqAdd, resAdd)	forward (reqAdd, resAdd)

ShowARCreated	SelectCalendar	doGet (reqPage, resPage)	-	forward (reqPage, resPage)	-
SelectAR-Webpage	AddVote	doGet (reqSelect, resSelect)	doGet (reqSelect, resSelect)	forward (reqSelect, resSelect)	forward (reqSelect, resSelect)
SelectAR-Webpage	AddDate	doGet (reqSelect, resSelect)	doGet (reqSelect, resSelect)	forward (reqSelect, resSelect)	forward (reqSelect, resSelect)
SelectAR-Webpage	SelectCalendar-Webpage	doGet(reqPage, resPage)	-	forward (reqPage, resPage)	-
AddVote	ShowVoted	doPost (reqVote, resVote)	doPost (reqVote, resVote)	forward (reqVote, resVote)	forward (reqVote, resVote)
ShowVoted	SelectCalendar-Webpage	doGet (reqPage, resPage)	-	forward (reqPage, resPage)	-
AddDate	ShowDate	doPost (reqDate, resDate)	doPost (reqDate, resDate)	forward (reqDate, resDate)	forward (reqDate, resDate)
ShowDate	SelectCalendar-Webpage	doGet (reqPage, resPage)	-	forward (reqPage, resPage)	-

Validation II

The state machine are consistent with the life-cycle model of Step A6:
All states are covered by a life-cycle:

Table 2.14: Validation II.I

GroupMemberGUI	
$LC_{GroupMember} = (showCalendar \mid addAppointment \mid AnswerAR)^*$	
State	**Covered by Life Cycle Part**
Init	addAppointment, AnswerAR, showCalendar
DefaultWebpage	addAppointment, AnswerAR, showCalendar
SelectAR-Webpage	AnswerAR
AddVote	AnswerAR
AddDate	AnswerAR
ShowDate	AnswerAR
ShowVoted	AnswerAR
SelectCalendar-Webpage	addAppointment, showCalendar
ShowCalendar	showCalendar
AddAR	addAppointment
ShowARCreated	addAppointment

All transitions are covered by a life-cycle:

Table 2.15: Validation II.II

Component GroupMemberGUI				
$LC_{GroupMember} = (showCalendar \mid addAppointment \mid AnswerAR)^*$				
Source State	**Target State**	**Input Signal**	**Output Signal**	**life cycle part**
Init	DefaultWebpage	doGet (reqDefault, resDefault)	forward (reqDefault, resDefault)	$(showCalendar \mid addAppointment \mid AnswerAR)^*$
DefaultWebpage	SelectCalendarWebpage	doGet (reqPage, resPage)	forward (reqPage, resPage)	$showCalendar \mid addAppointment \mid AnswerAR$
SelectARWebpage	AddVote	doGet (reqSelect, resSelect)	forward (reqSelect, resSelect)	AnswerAR
SelectARWebpage	AddDate	doGet (reqSelect, resSelect)	forward (reqSelect, resSelect)	AnswerAR
SelectARWebpage	SelectCalendarWebpage	doGet(reqPage, resPage)	forward (reqPage, resPage)	$(showCalendar \mid addAppointment \mid AnswerAR)^*$
AddVote	ShowVoted	doPost (reqVote, resVote)	forward (reqVote, resVote)	AnswerAR
ShowVoted	SelectARWebpage	doGet (reqPage, resPage)	forward (reqPage, resPage)	$AnswerAR)^*$
AddDate	ShowDate	doPost (reqDate, resDate)	forward (reqDate, resDate)	AnswerAR
ShowDate	SelectARWebpage	doGet (reqPage, resPage)	forward (reqPage, resPage)	$AnswerAR)^*$
SelectCalendarWebpage	DefaultWebpage	doGet (reqDefault, resDefault)	forward (reqDefault, resDefault)	$(showCalendar \mid addAppointment \mid AnswerAR)^*$
SelectCalendarWebpage	ShowCalendar	doPost (reqCal, resCal)	forward (reqCal, resCal)	showCalendar
ShowCalendar	SelectCalendarWebpage	doGet (reqPage, resPage)	forward (reqPage, resPage)	$showCalendar)^*$
SelectCalendarWebpage	AddAR	doGet (reqSelect, resSelect)	forward (reqSelect, resSelect)	addAppointment
SelectCalendarWebpage	SelectARWebpage	doGet (reqPage, resPage)	forward (reqPage, resPage)	AnswerAR
AddAR	ShowARCreated	doPost (reqAdd, resAdd)	forward (reqAdd, resAdd)	addAppointment
ShowARCreated	SelectCalendarWebpage	doGet (reqPage, resPage)	forward (reqPage, resPage)	$(showCalendar \mid addAppointment \mid AnswerAR)^*$

3 Glossary

Table 3.1: Glossary

Name	Type	Description	Source
A			
accessGC	phenomenon, auxilary function	GM accesses the Group Calendar	context diagram, problem diagram showCalendar, sdShowCalendar, class model
actual	attribute	represents if a participant is actual participant or not	Class model
Actual participant	biddable Domain	Group member who will attend the appointment	Requirements
addAppointment	name of sequence diagram	name of the squence diagram addAppointment and piece of the lifecycle	LC
AddAR	state predicate	The input form for creating an appointmentRequest is shown	sdAddAppointment_app
AddAR	state	Indicates that the input form for creating an appointment request is shown	state machine GroupMemberGUI
addDate	phenomenon	CA adding a date	context diagram, problem diagram AnswerAR, sdAnswerAR, sdAnswerAR_app
addDate	state predicate	The input form for adding a new date to the selected appointment request is shown	sdAnswerAR_app
AddDate	state	Indicates that input form for adding a new date to the selected appointment request is shown	state machine GroupMemberGUI
addGroup	phenomenon	CA adds Group to group Database	context diagramm
addMember	phenomenon	Member is added to Group	context diagram
AddressData	class	data type to represent an address	class model
addVote	phenomenon	Calendar App sets as possible/not possible for Group Member	context diagram, problem diagram AnswerAR, sdAnswerAR, sdAnswerAR_app
AddVote	state predicate	The input form for voting on the selected appointment request is shown	sdAnswerAR_app
AddVote	state	Indicates that the input form for voting on the selected appointment request is shown	state machine GroupMemberGUI

Table 3.1: Glossar

Name	Type	Description	Source
AnswerAR	name of sequence diagram	name of the squence diagram AnswerAR and piece of the lifecycle	LC
ApacheTomcat	connection domain	An Open Source JSP and servlet Container from the Apache Foundation.	TCD
APIProvided	class	Provided API that offers the current date	class model
Application	machine Domain	Collaborative Calendar App	Requirements
Appointments	phenomenon	the fixed Appointment	context diagram, problem diagram showCalendar, problem diagram FinalizeAppointment
AppointmentRequest	class	represents the webpage that the user interacts with	Class model
Appointment Request	connection Domain	a Request to create a new appointment, concreticizes interface feedbackGM	problem diagram AddAppointment, problem diagram AnswerAR, problem diagram showCalendar, sdShowCalendar, sdAnswerAR, sdAddAppointment, class model
appointmentRequest	phenomenon	Group Calendar sends Appointment Requests	context diagram, problem diagram AddAppointment, problem diagram AnswerAR, sdAddAppointment, sdAnswerAR
AppointmentExists	state predicate	The Appointment is created and exists	sdFinalizeAppointment
AppointmentData	message	reply message containing all AppointmentRequests and Appointments	sdFinalizeAppointment, sdShowCalendar
apr	message	the result of an query	sdShowCalendar_app
ar	object	object of the type ARData	sdShowCalendar_app
AR		Abbrevation of Appointment Request	context diagram
ARData	class	Represents an appointment request	class model
ARexists	state predicate	The AppointmentRequest is created and exists	sdAddAppointment, sdAnswerAR, sdFinalizeAppointment
AT		Abbreviation of Apache Tomcat	TCD
B			
C			
CA		Abbrevation of Calendar App	context diagram

Table 3.1: Glossar

Name	Type	Description	Source
CAAR		Abbrevation of CA_AddAR	problem diagram AddAppointment
CA_AAR		Abbrevation of CA_AnswerAR	problem diagram AddAppointment
CA_AddAR	class	represents the machine CA_AddAR	Class model
CA_AddAR	machine	Derivation of Calendar App that provides the functionality of adding an appointment request	problem diagram AddAppointment, sdAddApointment, class model
CA_AnswerAR	machine	Derivation of Calander App that provides the functionality of answering an appointment	problem diagram AnswerAR, sdAnswerAR, class model
CA_FinalizeRoutine	machine	Derivation of Calendar App that provides the routine to finalize appointments	problem diagram FinalizeAppointment
CA_FR		Abbrevation of CA_FinalizeRoutine	problem diagram FinalizeAppointment
CA_SC		Abbrevation of CA_showCal	problem diagram showCalendar
CA_showCal	machine	Derivation of Calendar App to provide the functionality to show the calendar	problem diagram showCalendar, sdShowCalendar
CA_FinalizeRoutine			problem diagram FinalizeAppointment, sdFinalizeAppointment
Calendar App	machine Domain	the actual Application	context diagram, TCD
CalendarData	connection Domain	the actual Data that the Calendar ontains, connects to CA_showCal, concretizes sendCalendarData	problem diagram showCalendar, sdShowCalendar
CD		Abbrevation of CalendarData	problem diagram showCalendar
changeRole	phenomenon	assign or remove roles from GM	context diagram
changedRole	phenomenon	CA changes role of Group Member in GD	context diagram
country	attribute	Represents the country where an Appointment take place	class model
create	message	creates an object of a defined type	sdShowCalendar_app
createAR	phenomenon	GM creates an Appointment Request	context diagram, problem diagram AddAppointment, sdAddAppointment, class model
createDate	phenomenon, auxilary function	a new date for an appointment request is created	context diagram, problem diagram AnswerAR, sdAnswerAR, class model, sdAnswerAR_app
createGroup	phenomenon	User creates a group	context diagram

Table 3.1: Glossar

Name	Type	Description	Source
createVote	phenomenon, auxiliary function	creates the vote from each Participant in the Calendar	problem diagram AnswerAR, sdAnswerAR, class model, sdAnswerAR_app
createdAR	phenomenon	Adding an appointment request to the Group Calendar with all possible dates	context diagram, problem diagram AddAppointment, sdAddAppointment, sdAddAppointment_app
createdUser	phenomenon	After registration a user is created in the User Database	context diagram
currentDate()	auxiliary function	Provides the current Date	class model
D			
date	attribute	represents a specified date	class model
date	phenomenon	return value of adding a date	sdAnswerAR_app
Dates	phenomenon, auxiliary function	CA returns the preliminary dates that are to vote	problem diagram AnswerAR, sdAnswerAR, class model
DateData	class	Represents a date submitted to an appointment request	class model
DateExists	state predicate	The Date is created and exists	sdAnswerAR
day	attribute	represents the day of a date	class model
deadline	attribute	represents the deadline when the appointment will be fixed	class model
DefaultWebpage	state	Indicates the starting page	state machine GroupMemberGUI
description	attribute	represents the description of a appointment request	class model
doGet	technical phenomenon	Defined in abstract class javax.servlet.http.HttpServlet	TCD
doGet	message	Java API function to request a HTML webpage	sdAddAppointment_app, sdAnswerAR_app, sdshowCalendar_app
doPost	technical phenomenon	Defined in abstract class javax.servlet.http.Http.Servlet	TCD
doPost	message	Java API function to send an input form to a HTML webpage	sdAddAppointment_app, sdAnswer_app
duration	attribute	represents the duration of the appointment	class model
E			
executeQuery	technical phenomenon	defined in interface java.sql.Statement	TCD
executeQuery	message	Java API function to send an SQL query command to a MYSQL Database	sdFinalizeRoutine_app, sdShowCalendar_app
executeUpdate	technical phenomenon	defined in interface java.sql.Statement	TCD

Table 3.1: Glossar

Name	Type	Description	Source
executeUpdate	message	Java API function to send an SQL update command to a MySQL database	sdAddAppointment_app, sdAnswerAR_app, sdFinalizeRoutine_app
F			
feedbackGM	phenomenon	sends feedback to the Group Member	context diagram
feedbackUser	phenomenon	sends feedback to the User whether their registration was successful or not	context diagram
fixed	attribute	represents the boolean value whether a date is fixed or not	class model
FinalizeAppointment	name of sequence diagram	name of the sequence diagram FinalizeAppointment and piece of the lifecycle	LC
forward	technical phenomenon	defined in interface javax.servlet.RequestDispatcher	TCD
forward	message	Java API function to send a message back to the users webpage	sdAddAppointment_app, sdAnswerAR_app, sdShowCalendar_app
G			
GA		Abbrevation of Group Administrator	context diagram
GC		Abbrevation of Group Calendar	context diagram
GD		Abbrevation of Group Database	context diagram
getDeadline	message, auxilary function	Returns the deadline of an AppointmentRequest	sdFinalizeAppointment, class model, sdFinalizeAppointment_app
getGroupData	phenomenon	allows the Group Member to get a List of all Group Members	context diagram
getUserData	phenomenon	allows the registered User to see their Data	context diagram
GM		Abbreviation of Group Member	context diagram
GMCmds	interface	used to trigger the operations from the group member	subArchAddAppointment, subArchAnswerAR, subArchShowCalendar, globalArch
Group	biddable Domain	A set of group members with access to the same calendar	Requirements
Groups	phenomenon	the Data about the existent Groups returned by the Group Database	context diagram
Group administrator	biddable Domain	Group member with additional permissions	Requirements, context diagram

Table 3.1: Glossar

Name	Type	Description	Source
Group Calendar	designed/lexical Domain	the Groups own Calendar, connects to CalendarData	context diagram, problem diagram addAppointment, problem diagram answerAR, problem diagram showCal, problem diagram FinalizeAppointment, sdShowCalendar, sdAddappointment, sdAnswerAR, sdFinalizeAppointment, class data
GroupCalendar	class	represents a group calendar with all appointments and appointment requests	class model
GroupCalendar_Adapter	component	responsible to create and maintain tables for all persistent classes	subArchAddAppointment, subArchAnswerAR, subArchShowCalendar, subArchFinalizeRoutine, globalArch
Group Database	designed/lexical Domain	The Database with information about the groups	context diagram
Group Member	biddable Domain	The role of a registered user within a group, connects to Calendar App	Requirements, problem diagram AddAppointment, problem diagram AnswerAR, problem diagram ShowCalendar, sdAnswerAR, sdShowCalendar, sdAddAppointment,TCD
GroupMemberWebBrowser	connection domain	Web browser used by group member, e.g. Chrome	TCD
GroupMemberGUI	component	web interface for group members	subArchAddAppointment, subArchAnswerAR, subArchShowCalendar, globalArch
gui	technical phenomenon	User interface of HTML webpages (defines by https://www.w3.org/TR/html5)	TCD
H			
handTo	attribute	represents the addrressant's name or room number	class model
hour	attribute	represents the hour of a date	class model
http	technical phenomenon	defined in Request for Comments(RFC) 2616, (Network Working Group, 1999)	TCD
I			
id	attribute	represents unique id of appointment request represents unique id of date represents unique id of participants	class Model

Table 3.1: Glossar

Name	Type	Description	Source
		represents unique id of group calendar	
Idle	state	Indicates that the server waits for incoming requests.	State Machine GroupMemberGUI
IGroupCalendar	interface	used to trigger the operations from the machine to the database	subArchAddAppointment, subArchAnswerAR, subArchShowCalendar, subArchFinalizeRoutine, globalArch
inviteMember	phenomenon	GA invites a RU to a group	context diagram
isEarlier	auxilary function	checks whether a date is earlier then the current date	
ITimer	interface	used to trigger the internal operation "getDeadline" periodically	subArchFinalizeRoutine, globalArch
J			
joinGroup	phenomoneon	RU joins a Group	context diagram
K			
L			
$LC_{GroupMember}$	life-cycle	Life-cycle for one group member	LC
$LC_{CalendarApp}$	life-cycle	Combined life-cycle(all groupmembers and the internal operation)	LC
location	attribute	represents the place where the appointment will take place	class model
login	phenomenon	user logs in	context diagram
logout	phenomenon	user logs out	context diagram
M			
makeAR	phenomenon, auxiliary function	creates the Appointment Request	problem diagram AddAppointment, sdAddAppointment, class model, sdAddAppointment_app
markDates	phenomenon	marks dates as either preliminary or fixed	context diagram, problem diagram FinalizeAppointment, sdFinalizeAppointment, sdFinalizeRoutine_app
minute	attribute	represents a minute of a date	class model
month	attribute	represents a month of a date	class model
N			
name	attribute	represents the name of an appointment request	class model
O			
okRepresentation	phenomenon	feedback to the Group Member	problem diagram AddAppointment, sdAddAppointment
Ok	phenomenon	return value	sdAddAppointment_app
P			

Table 3.1: Glossar

Name	Type	Description	Source
P		Abbrevation of Participant	context diagram
Participants	class	Represents a participant	class model
Planned participant	biddable Domain	Group member meant to attend an appointment	Facts
Q			
queryRes	message	the result of a query to the SQL database	sdShowCalendar_app
R			
register	phenomenon	user creates and Account	context diagram
Registered user	biddable Domain	A User who created an account	Requirements, context diagram
removeMember	phenomenon	removes a Member from the Group	context diagram
removedMember	phenomenon	the Calendar App forwards removeMember	context diagram
removePreDate	phenomenon	removes a prelimanary date	context diagram, problem diagram FinalizeAppointment, sdFinalizeAppointment, sdFinalizeRoutine_app
requestCal	phenomenon	requests the AppointmentData from the GroupCalendar	problem diagram showCalendar, sdShowCalendar, sdShowCalendar_app
requestCD	phenomenon, auxilary function	forwards the access request of the Group Member to the machine	problem diagram showCalendar, sdShowCalendar, sdShowCalendar_app, class model
RU		Abbreviation of Registered User	context diagram
rUsers	phenomenon	all registered users in the User Database	context diagram
S			
sendCD	phenomenon, auxilary function	provides the requested CalendarData	problem diagram showCalendar, sdShowCalendar, class model
showARCreated	state predicate	The confirmation of the creation of the appointment is shown	sdAddAppointment_app
ShowARCreated	state	Indicates that the confirmation of the creation of the appointment request is shown	state machine GroupMemberGUI
ShowCalendar	state predicate	The data of the Group Calendar are shown	sdShowCalnedar_app
showCalendar	name of sequence diagram	name of the squence diagram showCalendar and piece of the lifecycle	LC
showDate	state predicate	The confirmation of the addition of a new date to a appointment request is shown.	sdAnswerAR_app

Table 3.1: Glossar

Name	Type	Description	Source
ShowDate	state	Indicates that the confirmation of the addition of a date to a appointment request is shown	state machine GroupMemberGUI
showDates	phenomenon	AR sends preliminary dates to GM	problem diagram AnswerAR, sdAnswerAR
showVoted	state predicate	The confirmation of the voting is shown	sdAnswerAR_app
ShowVoted	state	Indicates that the confirmation of the voting is shown	state machine GroupMemberGUI
showVotes	phenomenon	shows the number of votes for each date	problem diagram AnswerAR, sdAnswerAR
SelectARWebpage	state predicate	The requested appointment request is displayed. SelectARWebpage $->$ ARExists	sdAnswerAR_app
SelectARWebpage	state	Indicates the overview to respond on an appointment request	state machine GroupMemberGUI
SelectCalendarWebpage	state predicate	The requested Group Calendar is displayed.	sdAddAppointment_app, sdShowCalendar_app
SelectCalendarWebpage	state	Indicates the overview of the selected group calendar	state machine GroupMemberGUI
sendGroupData	phenomenon	Calendar App sends list Group Members	context diagram
sendUserData	phenomenon	Calendar App sends Registered Users Information	context diagram
sendInvite	phenomenon	CA sends a group invite	context diagram
sendCalendarData	phenomenon	return value provided by the Calendar App if the Group Member requests the Calendar data	context diagram, problem diagram showCalendar, sdShowCalendar
setParticipants	phenomenon	marks a Group Member as an actual Participant	context diagram, problem diagram FinalizeAppointment, sdFinalizeAppointment, sdFinalizeRoutine
suggestDate	phenomenon, auxiliary function	Group member suggests a date for the appointment	context diagram, problem diagram AnswerAR, sdAnswerAR, class model
Sessions	phenomenon	time while a Registered User is logged in	context diagram
setSession	phenomenon	sets the Users session in the User Database	context diagram
ShowCalendar	state predicate	The dates of the calendar are shown	sdShowCalendar_app
ShowCalendar	state	Indicates that the dates of the calendar are shown	state machine GroupMemberGUI
showOk	phenomenon, auxiliary function	CAAR confirms AppointmentRequest as successful	problem diagram AddAppointment, sdAddAppointment, class model

Table 3.1: Glossar

Name	Type	Description	Source
SQLDatabase	causal domain	An database using SQL commands	TCD
street	attribute	represents the street in AddressData	class model
T			
TimeData	class	data type that represents a date with minutes	class model
Timer	reusedComponent	given component initiating the internal operation "getDeadline"	subArchFinalizeRoutine, globalArch
town	attribute	represents the town in AddressData	class model
U			
U		Abbrevation of User	context diagram
UD		Abbrevation of User Database	context diagram
User	biddable Domain	User of the system	Requirements
User Database	designed/lexical Domain	A Database that contains all information of the registered users	context diagram
V			
vote	phenomenon, auxiliary function	group member selects whether a date is possible	context diagram, problem diagram AnswerAR, class model, sdAnswerAR
votes	phenomenon, auxiliary function	CA_AAR sends votes of Group Members for a date, return value	problem diagram AnswerAR, class model, sdAnswerAR, sdAnswerAR_app
VoteExists	state predicate	The Vote is created and exists	sdAnswerAR
W			
X			
Y			
year	attribute	represents a year of a date	class model
Z			
zip	attribute	represents a zip code in AddressData	class model