Collaborative Calendar App

Software Engineering Lab WS 20/21

Salwan Allahham (3074803), Mohammed Al Kabbani (3077146), Mohamad Moussa (3078672), Ning Jiahao (3100085), Kawa Khuder (3085018), Qishuang Li (3100089).

Lab Time: Don 12 - 14

Supervisor: Lirijan Sabani

Date of submission: 18.02.2021

Requirements:

- R1: Every registered user can create a group and becomes an administrator of this group.
- R2: Registered users can join the groups they are invited to.
- R3: Group administrators can also change the rights of group members (including other group administrators) to group administrator, group member, or nonmember.
- R4: Every group member can create a new appointment request in the group's calendar.
- R5: All dates at which the appointment may take place shall be added to the group calendar and marked as preliminary appointment dates.
- R6: 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. All other preliminary dates of the appointment are removed from the calendar.
- R7: The chosen date is added as fixed date of the appointment to the calendar and all other preliminary dates of the appointment are removed from it. Additionally, all planned participants for whom the date is possible are recorded as actual participants.
- R8: Every group member can suggest dates for every appointment and 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.
- R9: All group members can access the group calendar containing all appointments of the group.
- R10: Group administrators can invite other registered users to become a member of the group.
- R11: To use the web application users have to register.
- R12: The user can login and logout.

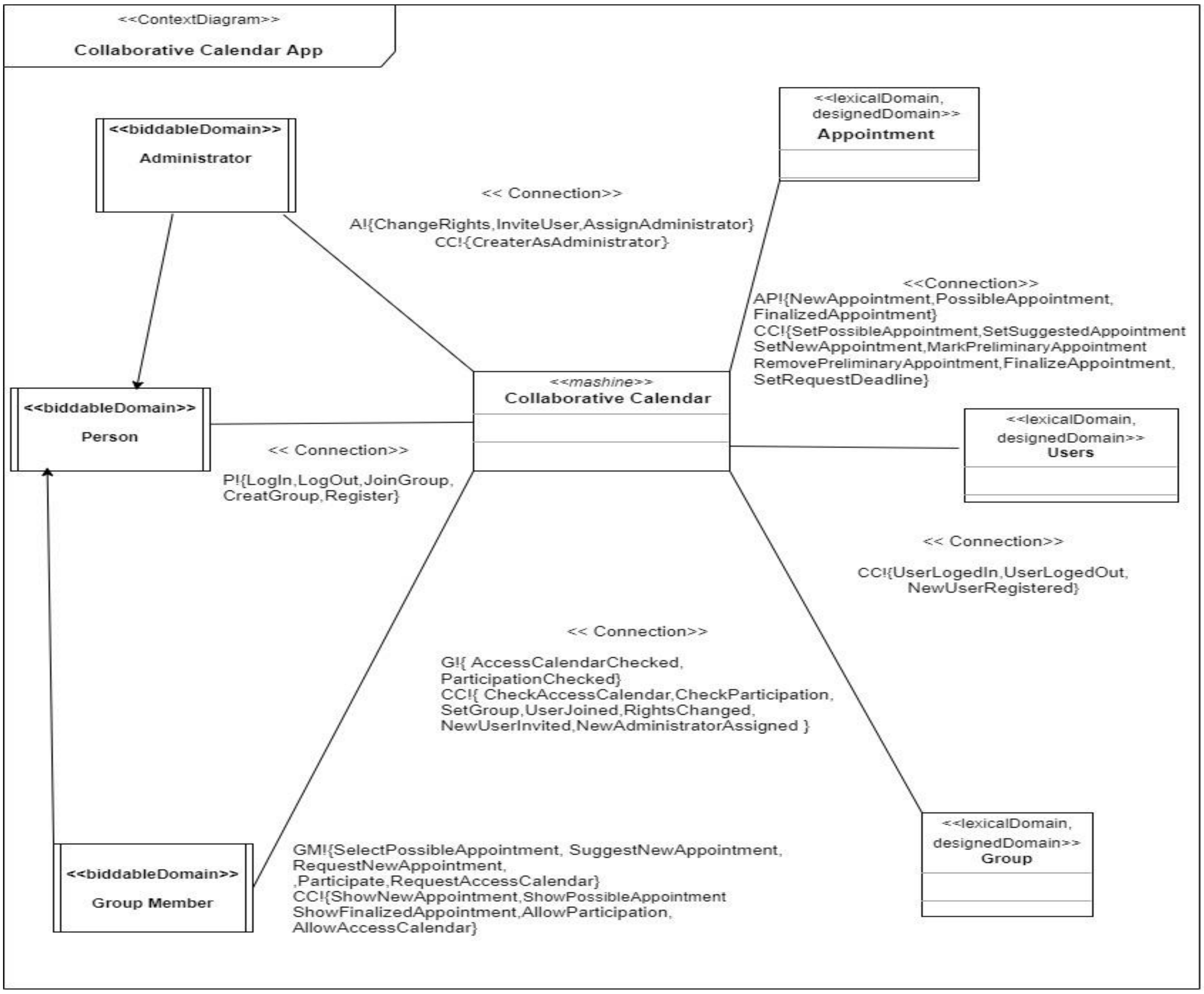
Assumptions:

- A1: A web application is suitable to be used on different platforms including mobile devices.
- A2: Registered users regularly check whether they were invited to groups.
- A3: One calendar per group is sufficient.
- A4: All group members check regularly whether new appointment requests exist they shall participate in.
- A5: The other planned participants can also suggest alternative appointments, but they do not have to if they selected at least one of the existing dates as possible.
- A6: The planned participants have to select which of the suggested dates are possible for them and which not.

- A7: If none of the suggested dates is possible for a planned participant, then he/she has to suggest alternative dates (at least one).
- A8: If new dates are added to an appointment, all planned participants have again to decide whether the newly added appointments are possible for them or not.
- A9: It is assumed that it is possible to have the appointment even if not all of the planned participants can participate.
- **Facts:**

- F1: Group administrators are special group members, and group members are special registered users.
- F2: Every appointment request has a name, description, location, duration, list of planned participants (group members), a list of dates at which the appointment may take place, and a deadline after which the calendar app will automatically select an appropriate date for the appointment.

The Context Diagram:



<<ContextDiagram>>
Collaborative Calendar App
<<biddableDomain>>
Administrator
<<lexicalDomain, designedDomain>>
Appointment
<< Connection>>
A!{ChangeRights,InviteUser,AssignAdministrator}
CC!{CreaterAsAdministrator}
<<Connection>>
AP!{NewAppointment,PossibleAppointment, FinalizedAppointment}
CC!{SetPossibleAppointment,SetSuggestedAppointment SetNewAppointment,MarkPreliminaryAppointment RemovePreliminaryAppointment,FinalizeAppointment, SetRequestDeadline}
<<biddableDomain>>
Person
<<mashine>>
Collaborative Calendar
<<lexicalDomain, designedDomain>>
Users
<< Connection>>
P!{LogIn,LogOut,JoinGroup, CreatGroup,Register}
<< Connection>>
CC!{UserLogedIn,UserLogedOut, NewUserRegistered}
<< Connection>>
G!{ AccessCalendarChecked, ParticipationChecked}
CC!{ CheckAccessCalendar,CheckParticipation, SetGroup,UserJoined,RightsChanged, NewUserInvited,NewAdministratorAssigned }
<<biddableDomain>>
Group Member
GM!{SelectPossibleAppointment, SuggestNewAppointment, RequestNewAppointment, ,Participate,RequestAccessCalendar}
CC!{ShowNewAppointment,ShowPossibleAppointment ShowFinalizedAppointment,AllowParticipation, AllowAccessCalendar}
<<lexicalDomain, designedDomain>>
Group

The Validation

- The glossary contains the notions used in R and D.
- The notions mentioned in R and D are contained in the glossary
- Domains and phenomena of the context diagram are consistent with R and D.
- There is only one Context Diagram
- The machine domain must control at least one interface.

Notion in CD	Notion in R/D	Type
Administrator	administrator	domain
RequestAccessCalendar, CheckAccessCalender, AccessCalendarChecked, AllowAccessCalendar	All group members can access the group calendar	phenomenon
Participate,CheckParticipation, ParticipationChecked, AllowParticipation	all planned participants for whom the date is possible are recorded as actual participants	phenomenon
Appointment	appointment	domain
CreaterAsAdministrator, NewAdministratorAssigned	becomes an administrator of this group.	phenomenon
ChangeRights,RightsChanged	Group administrators can also change the rights of group members	phenomenon
Collaborative Calendar	Collaborative Calendar App	machine
CreatGroup, SetGroup	Every registered user can create a group	phenomenon
Group	group	domain
NewUserInvited, inviteUser,	Group administrators can invite other registered users	phenomenon
LogIn,UserLogedOut	The user can login	phenomenon
LogOut,UserLogedOut	The user can logout	phenomenon
JoinGroup, UserJoined	users can join the groups	phenomenon
Register,NewUserRegistered	users have to register	phenomenon
FinalizeAppointment, FinalizedAppointment, ShowFinalizedAppointment,	The chosen date is added as fixed date	phenomenon
RequestNewAppointment, SetNewAppointment, NewAppointmentShowNewAppointment	create a new appointment request	phenomenon
SetRequestDeadline	If the configured deadline date of an appointment request	phenomenon
SelectPossibleAppointment, SetPossibleAppiontment, PossibleAppointment	all planned participants for whom the date is possible are recorded	phenomenon
SuggestNewAppointment SetSuggestedAppiontment	Every group member can suggest dates	phenomenon
Users	users	domain
Group member	Group member	domain
Person	Person	domain

- A context diagram has at least one machine domain.

  - Collaborative Calendar is the machine domain.

- Biddable domains cannot be directly connected to lexical domains.

  - No biddable domain is connected to a lexical domain.

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

  - Administrator, Person and Group Member are biddable domains.

Domain	Domain Type(s)	Connected Domain(s)	Connected Domain(s) Type(s)
Collaborative Calendar	machine domain	Administrator	Biddable domain
		Users	Lexical domain
		Appointment	Lexical domain
		Group	Lexical domain
		Person	Biddable domain
		Group member	Biddable domain
Administrator	Biddable domain	Collaborative Calendar	machine domain
		Person	Biddable domain
Users	Lexical domain	Collaborative Calendar	machine domain
Appointment	Lexical domain	Collaborative Calendar	machine domain
Group	Lexical domain	Collaborative Calendar	machine domain
Group member	Biddable domain	Collaborative Calendar	machine domain
		Person	Biddable domain
Person	Biddable domain	Collaborative Calendar	machine domain
		Group member	Biddable domain
		Administrator	Biddable domain

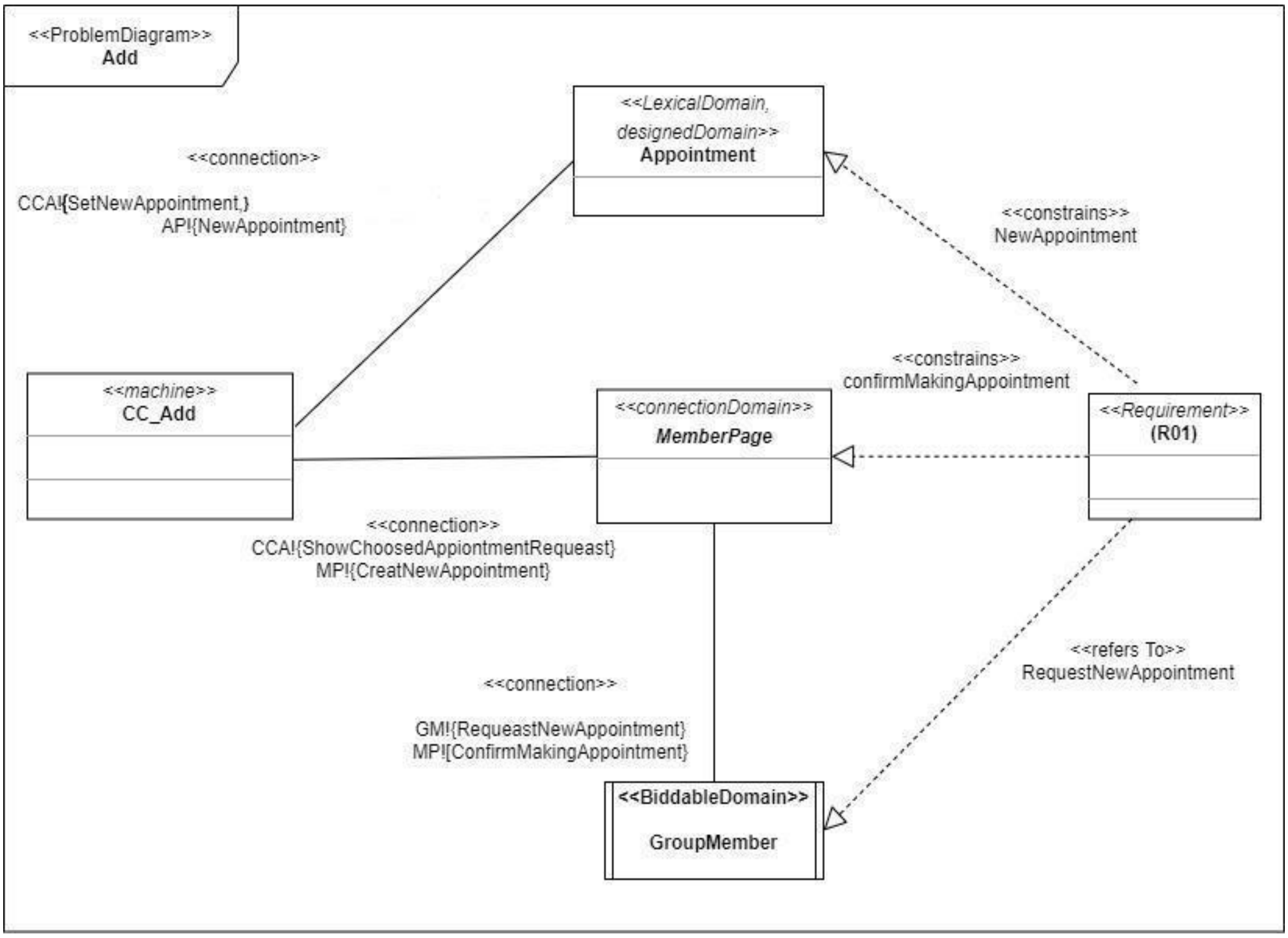
- Phenomena controlled by a biddable domain all have counterpart phenomena located between machine and causal/lexical/designed domains.

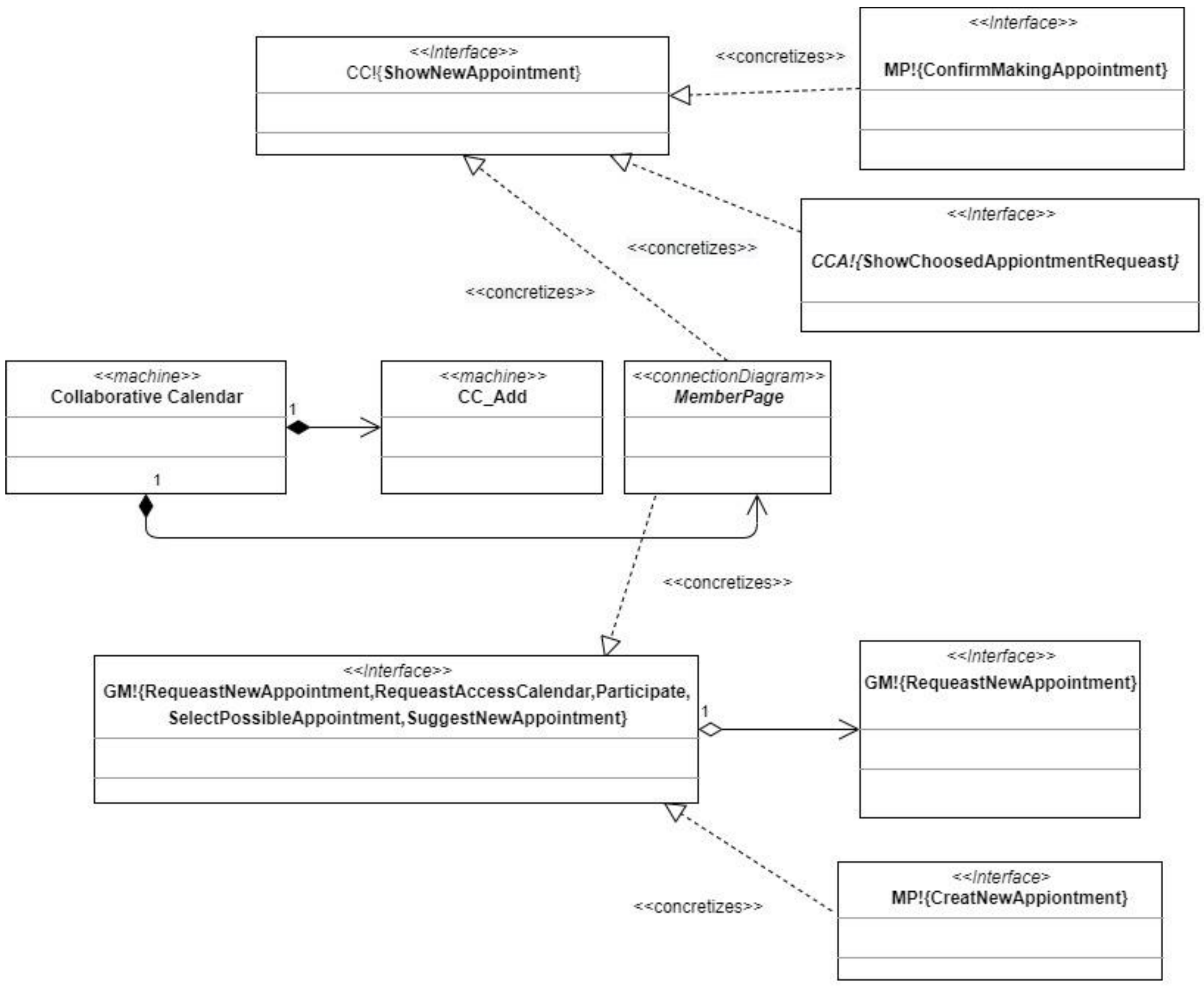
	biddable domain phenomena	**counterpart**
Person		
	LogIn,LogOut	UserLogedIn, UserLogedOut
	GreatGroup	SetGroup
	Register	NewUserRegistered
	JoinGroup	UserJoined
Administrator		
	ChangeRights	RightsChanged
	InviteUser	NewUserInvited
	AssignAdministrator	NewAdministratorAssigned
Group member		
	RequestNewAppointment	SetNewAppointment
	RequestAccessCalendar	CheckAccessCalendar
	SelectPossibleAppointment	SetPossibleAppointment
	SuggestNewAppointment	SetSuggestedAppointment
	Participate	CheckParticipation

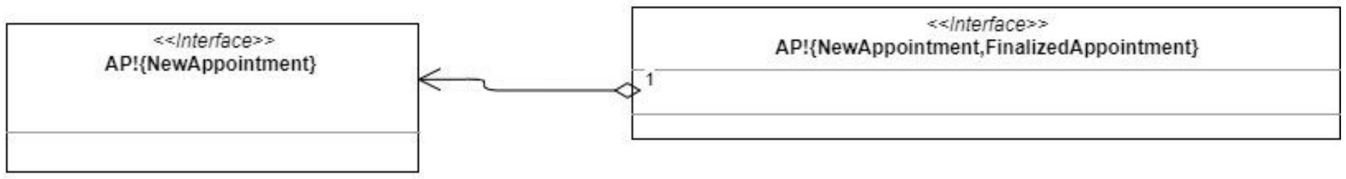
- Connection domains must have at least one observed and one controlled interface.
- For each phenomenon controlled by a connection domain, there must be at least one phenomenon controlled by one of the connected domains, i.e. observed by the connection domain.
- For each phenomenon observed by a connection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

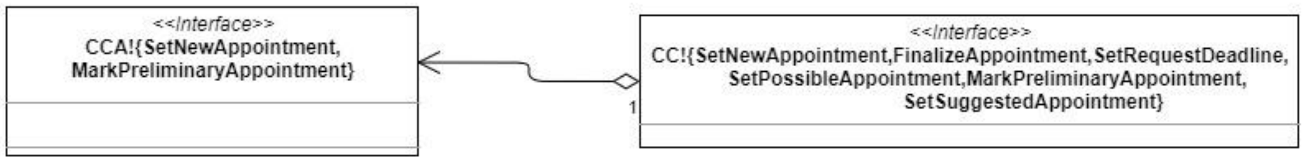
R01: Add Appointment (R4+R5)

Every group member can create a new appointment request in the group's calendar.
All dates at which the appointment may take place shall be added to the group calendar and marked as preliminary appointment dates.



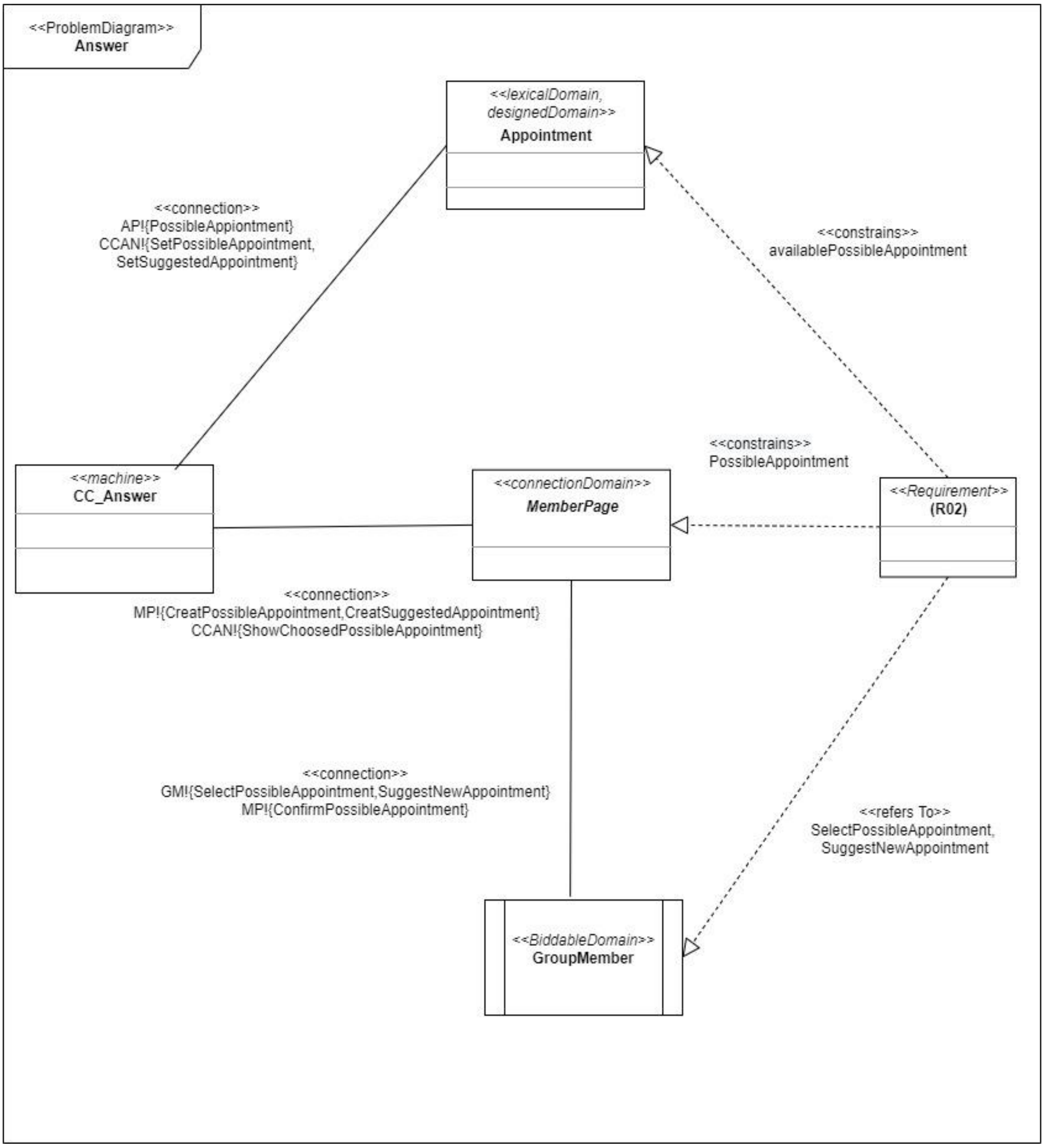

<<Interface>>
CC!{ShowNewAppointment}
<<concretizes>>
<<Interface>>
MP!{ConfirmMakingAppointment}
<<concretizes>>
<<Interface>>
CCA!{ShowChoosedAppiontmentRequeast}
<<concretizes>>
<<machine>>
Collaborative Calendar
1
1
<<machine>>
CC_Add
<<connectionDiagram>>
MemberPage
<<concretizes>>
<<Interface>>
GM!{RequeastNewAppointment,RequeastAccessCalendar,Participate,
SelectPossibleAppointment,SuggestNewAppointment}
1
<<Interface>>
GM!{RequeastNewAppointment}
<<concretizes>>
<<Interface>
MP!{CreatNewAppiontment}


<<Interface>>
AP!{NewAppointment}
<<Interface>>
AP!{NewAppointment,FinalizedAppointment}
1

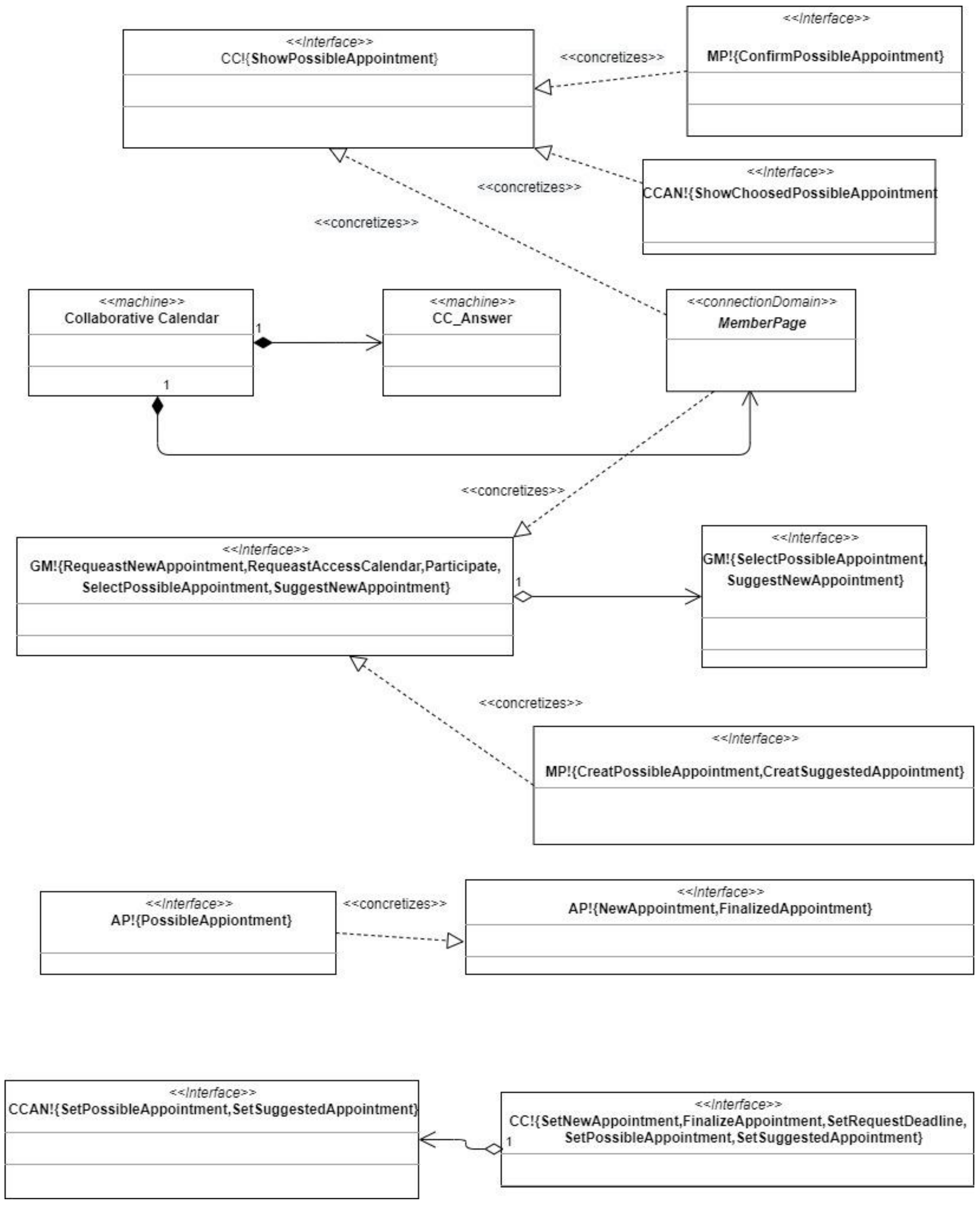

<<Interface>>
CCA!{SetNewAppointment,
MarkPreliminaryAppointment}
<<Interface>>
CC!{SetNewAppointment,FinalizeAppointment,SetRequestDeadline,
SetPossibleAppointment,MarkPreliminaryAppointment,
SetSuggestedAppointment}
1

R02: Answer Appointment(R8)

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.



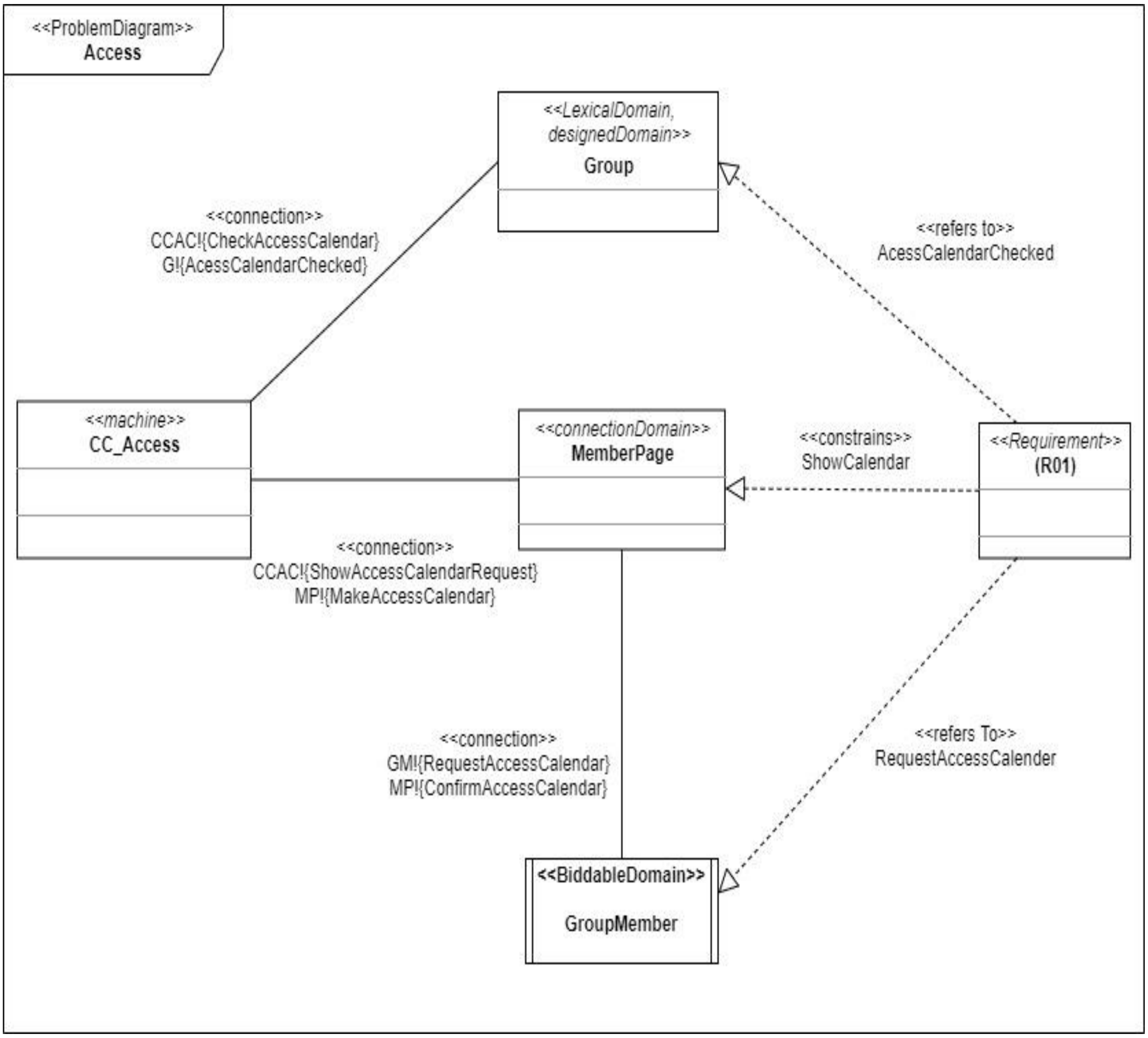
<<ProblemDiagram>>
Answer
<<lexicalDomain, designedDomain>>
Appointment
<<connection>>
AP!{PossibleAppiontment}
CCAN!{SetPossibleAppointment, SetSuggestedAppointment}
<<constrains>>
availablePossibleAppointment
<<constrains>>
PossibleAppointment
<<machine>>
CC_Answer
<<connectionDomain>>
MemberPage
<<Requirement>>
(R02)
<<connection>>
MP!{CreatPossibleAppointment,CreatSuggestedAppointment}
CCAN!{ShowChoosedPossibleAppointment}
<<connection>>
GM!{SelectPossibleAppointment,SuggestNewAppointment}
MP!{ConfirmPossibleAppointment}
<<refers To>>
SelectPossibleAppointment,
SuggestNewAppointment
<<BiddableDomain>>
GroupMember

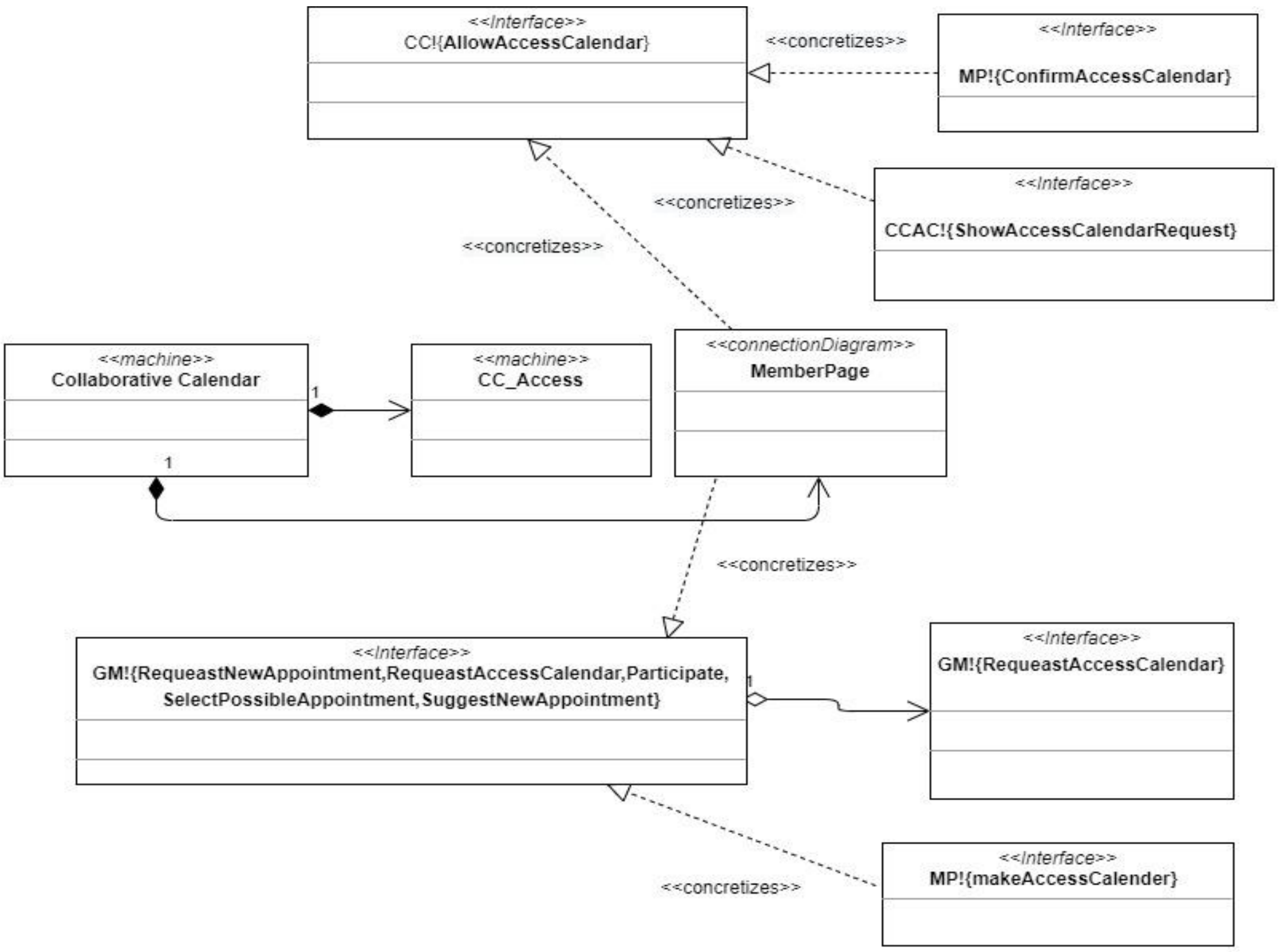


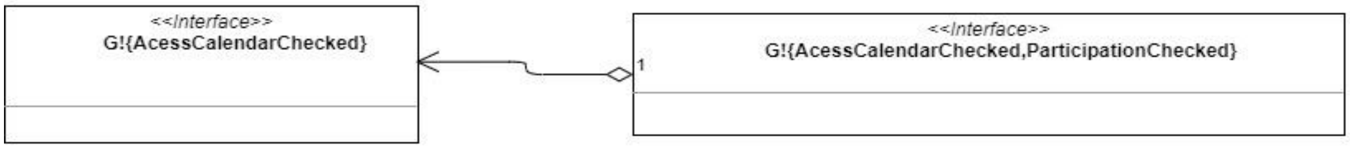
<<Interface>>
CC!{ShowPossibleAppointment}
<<concretizes>>
<<Interface>>
MP!{ConfirmPossibleAppointment}
<<concretizes>>
<<Interface>>
CCAN!{ShowChoosedPossibleAppointment
<<concretizes>>
<<machine>>
Collaborative Calendar
1
<<machine>>
CC_Answer
<<connectionDomain>>
MemberPage
1
<<concretizes>>
<<Interface>>
GM!{RequeastNewAppointment,RequeastAccessCalendar,Participate,
SelectPossibleAppointment,SuggestNewAppointment}
1
<<Interface>>
GM!{SelectPossibleAppointment,
SuggestNewAppointment}
<<concretizes>>
<<Interface>>
MP!{CreatPossibleAppointment,CreatSuggestedAppointment}
<<Interface>>
AP!{PossibleAppiontment}
<<concretizes>>
<<Interface>>
AP!{NewAppointment,FinalizedAppointment}
<<Interface>>
CCAN!{SetPossibleAppointment,SetSuggestedAppointment}
<<Interface>>
CC!{SetNewAppointment,FinalizeAppointment,SetRequestDeadline,
SetPossibleAppointment,SetSuggestedAppointment}
1

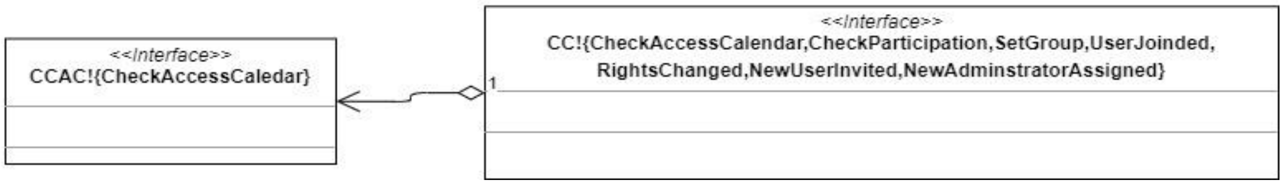
R03: Show Calendar(R9)

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





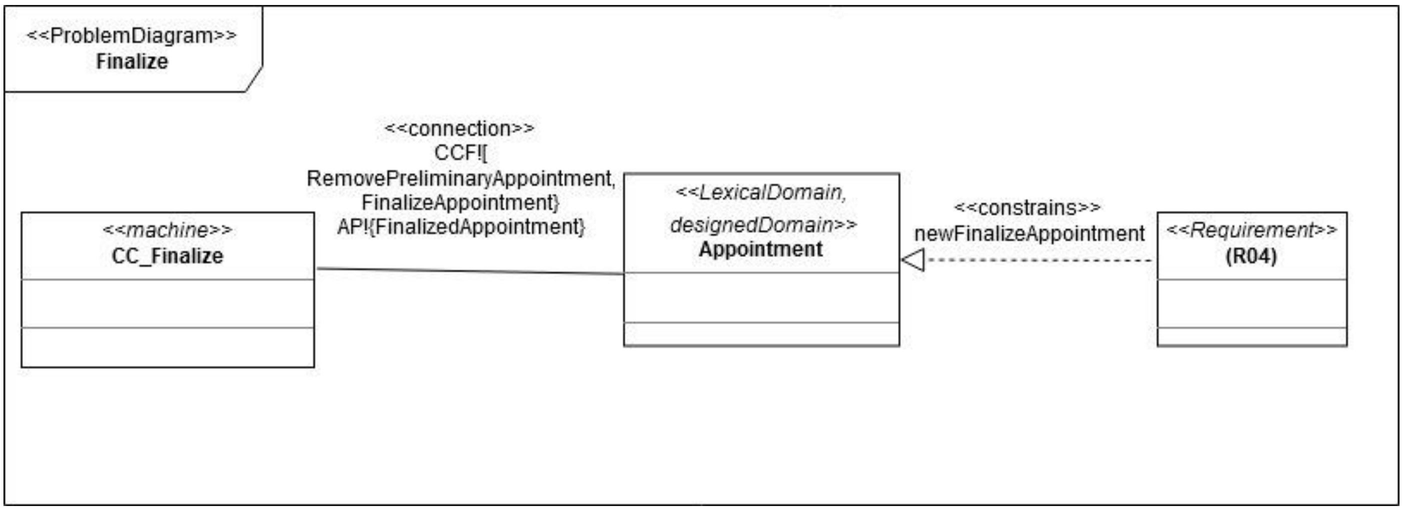




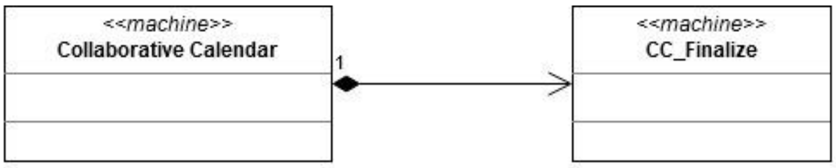
<<Interface>>
CCAC!{CheckAccessCaledar}
1
<<Interface>>
CC!{CheckAccessCalendar,CheckParticipation,SetGroup,UserJoinded,
RightsChanged,NewUserInvited,NewAdminstratorAssigned}

R04: Finalize Appointment (R5+R7)

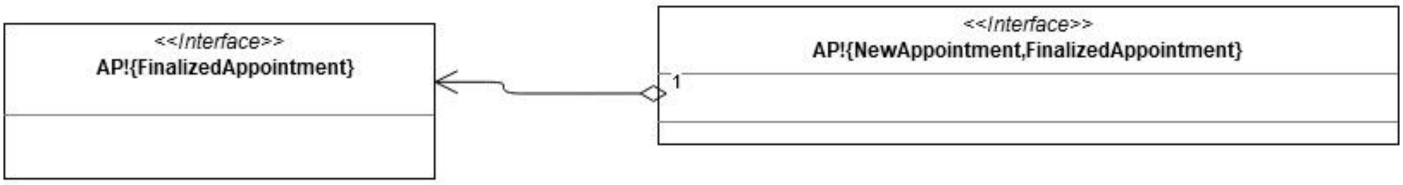
All dates at which the appointment may take place shall be added to the group calendar and marked as preliminary appointment dates.
The chosen date is added as fixed date of the appointment to the calendar and all other preliminary dates of the appointment are removed from it. Additionally, all planned participants for whom the date is possible are recorded as actual participant.



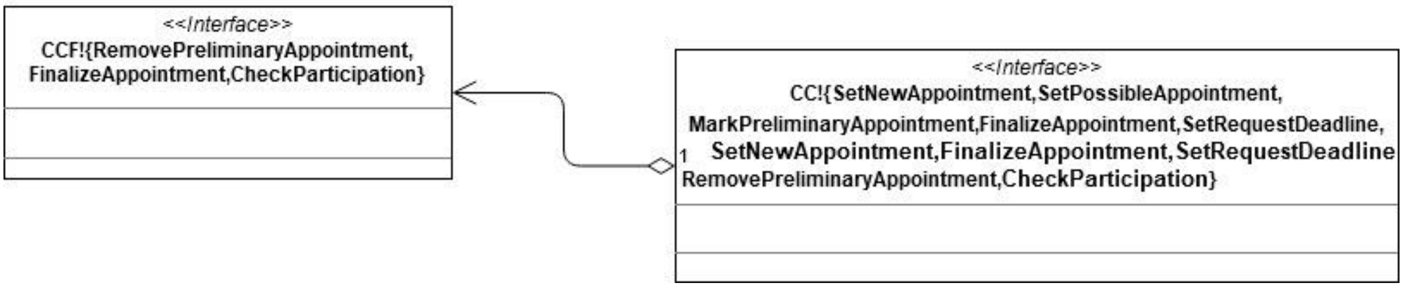
<<ProblemDiagram>>
Finalize
<<connection>>
CCF![
RemovePreliminaryAppointment,
FinalizeAppointment}
AP!{FinalizedAppointment}
<<machine>>
CC_Finalize
<<LexicalDomain,
designedDomain>>
Appointment
<<constrains>>
newFinalizeAppointment
<<Requirement>>
(R04)



<<machine>>
Collaborative Calendar
1
<<machine>>
CC_Finalize



<<Interface>>
AP!{FinalizedAppointment}
<<Interface>>
AP!{NewAppointment,FinalizedAppointment}
1



<<Interface>>
CCF!{RemovePreliminaryAppointment,
FinalizeAppointment,CheckParticipation}
<<Interface>>
CC!{SetNewAppointment,SetPossibleAppointment,
MarkPreliminaryAppointment,FinalizeAppointment,SetRequestDeadline,
SetNewAppointment,FinalizeAppointment,SetRequestDeadline
RemovePreliminaryAppointment,CheckParticipation}
1

The Validation:

- All requirements R are covered in some subproblem.

Requirement	**Covered in**	**Contained domain**	**domain type**	**constrained**	**controlled phenomena**
R01	pdAdd	CC_Add	machine		SetNewAppointment, ShowChoosedAppiontmentRequeast
		MemberPage	connection	X	CreatNewAppointment, ConfirmMakingAppointment
		Appointment	Lexical, designed	X	NewAppointment
		GroupMember	Biddable		RequeastNewAppointment
R02	pdAnswer	CC_Answer	machine		SetPossibleAppointment, SetSuggestedAppointment ShowChoosedPossibleAppointment
		MemberPage	connection	X	ConfirmPossibleAppointment CreatPossibleAppointment, CreatSuggestedAppointment
		Appointment	Lexical, designed	X	PossibleAppiontment
		GroupMember	Biddable		SelectPossibleAppointment, SuggestNewAppointment
R03	pdAccess	CC_Access	machine		CheckAccessCalendar ShowAccessCalendarRequest
		MemberPage	connection	X	MakeAccessCalendar ConfirmAccessCalendar
		Group	Lexical, designed		AcessCalendarChecked
		GroupMember	Biddable		RequestAccessCalendar
R04	pdFinalize	CC_Finalize	machine		RemovePreliminaryAppointment, FinalizeAppointment, CheckParticipation
		Appointment	Lexical, designed	x	FinalizedAppointment

- A problem diagram has exactly one machine domain.

Requirement	**Covered in**	**Contained domain**	**domain type**	**constrained**	**controlled phenomena**
R01	pdAdd	CC_Add	machine		SetNewAppointment, ShowChoosedAppiontmentRequeast
		MemberPage	connection	X	CreatNewAppointment, ConfirmMakingAppointment
		Appointment	Lexical, designed	X	NewAppointment
		GroupMember	Biddable		RequeastNewAppointment
R02	pdAnswer	CC_Answer	machine		SetPossibleAppointment, SetSuggestedAppointment ShowChoosedPossibleAppointment
		MemberPage	connection	X	ConfirmPossibleAppointment CreatPossibleAppointment, CreatSuggestedAppointment
		Appointment	Lexical, designed	X	PossibleAppiontment
		GroupMember	Biddable		SelectPossibleAppointment, SuggestNewAppointment
R03	pdAccess	CC_Access	machine		CheckAccessCalendar ShowAccessCalendarRequest
		MemberPage	connection	X	MakeAccessCalendar ConfirmAccessCalendar
		Group	Lexical, designed		AcessCalendarChecked
		GroupMember	Biddable		RequestAccessCalendar
R04	pdFinalize	CC_Finalize	machine		RemovePreliminaryAppointment, FinalizeAppointment, CheckParticipation
		Appointment	Lexical, designed	x	FinalizedAppointment

- A problem diagram contains at least one requirement.

Requirement	Covered in	Contained domain	domain type	constrained	controlled phenomena
R01	pdAdd	CC_Add	machine		SetNewAppointment, ShowChoosedAppiontmentRequeast
		MemberPage	connection	X	CreatNewAppointment, ConfirmMakingAppointment
		Appointment	Lexical, designed	X	NewAppointment
		GroupMember	Biddable		RequeastNewAppointment
R02	pdAnswer	CC_Answer	machine		SetPossibleAppointment, SetSuggestedAppointment ShowChoosedPossibleAppointment
		MemberPage	connection	X	ConfirmPossibleAppointment CreatPossibleAppointment, CreatSuggestedAppointment
		Appointment	Lexical, designed	X	PossibleAppiontment
		GroupMember	Biddable		SelectPossibleAppointment, SuggestNewAppointment
R03	pdAccess	CC_Access	machine		CheckAccessCalendar ShowAccessCalendarRequest
		MemberPage	connection	X	MakeAccessCalendar ConfirmAccessCalendar
		Group	Lexical, designed		AcessCalendarChecked
		GroupMember	Biddable		RequestAccessCalendar
R04	pdFinalize	CC_Finalize	machine		RemovePreliminaryAppointment, FinalizeAppointment CheckParticipation
		Appointment	Lexical, designed	x	FinalizedAppointment

- The machine domain must control at least one interface.

Requirement	**Covered in**	**Contained domain**	**domain type**	**constrained**	**controlled phenomena**
R01	pdAdd	CC_Add	machine		SetNewAppointment, ShowChoosedAppiontmentRequeast
		MemberPage	connection	X	CreatNewAppointment, ConfirmMakingAppointment
		Appointment	Lexical, designed	X	NewAppointment
		GroupMember	Biddable		RequeastNewAppointment
R02	pdAnswer	CC_Answer	machine		SetPossibleAppointment, SetSuggestedAppointment ShowChoosedPossibleAppointment
		MemberPage	connection	X	ConfirmPossibleAppointment CreatPossibleAppointment, CreatSuggestedAppointment
		Appointment	Lexical, designed	X	PossibleAppiontment
		GroupMember	Biddable		SelectPossibleAppointment, SuggestNewAppointment
R03	pdAccess	CC_Access	machine		CheckAccessCalendar ShowAccessCalendarRequest
		MemberPage	connection	X	MakeAccessCalendar ConfirmAccessCalendar
		Group	Lexical, designed		AcessCalendarChecked
		GroupMember	Biddable		RequestAccessCalendar
R04	pdFinalize	CC_Finalize	machine		RemovePreliminaryAppointment, FinalizeAppointment CheckParticipation
		Appointment	Lexical, designed	x	FinalizedAppointment

- Requirements constrain at least one domain.

Requirement	**Covered in**	**Contained domain**	**domain type**	**constrained**	**controlled phenomena**
R01	pdAdd	CC_Add	machine		SetNewAppointment, ShowChoosedAppiontmentRequeast
		MemberPage	connection	X	CreatNewAppointment, ConfirmMakingAppointment
		Appointment	Lexical, designed	X	NewAppointment
		GroupMember	Biddable		RequeastNewAppointment
R02	pdAnswer	CC_Answer	machine		SetPossibleAppointment, SetSuggestedAppointment ShowChoosedPossibleAppointment
		MemberPage	connection	X	ConfirmPossibleAppointment CreatPossibleAppointment, CreatSuggestedAppointment
		Appointment	Lexical, designed	X	PossibleAppiontment
		GroupMember	Biddable		SelectPossibleAppointment, SuggestNewAppointment
R03	pdAccess	CC_Access	machine		CheckAccessCalendar ShowAccessCalendarRequest
		MemberPage	connection	X	MakeAccessCalendar ConfirmAccessCalendar
		Group	Lexical, designed		AcessCalendarChecked
		GroupMember	Biddable		RequestAccessCalendar
R04	pdFinalize	CC_Finalize	machine		RemovePreliminaryAppointment, FinalizeAppointment CheckParticipation
		Appointment	Lexical, designed	x	FinalizedAppointment

- Requirements do not constrain machine(s).

Requirement	**Covered in**	**Contained domain**	**domain type**	**constrained**	**controlled phenomena**
R01	pdAdd	CC_Add	machine		SetNewAppointment, ShowChoosedAppiontmentRequeast
		MemberPage	connection	X	CreatNewAppointment, ConfirmMakingAppointment
		Appointment	Lexical, designed	X	NewAppointment
		GroupMember	Biddable		RequeastNewAppointment
R02	pdAnswer	CC_Answer	machine		SetPossibleAppointment, SetSuggestedAppointment ShowChoosedPossibleAppointment
		MemberPage	connection	X	ConfirmPossibleAppointment CreatPossibleAppointment, CreatSuggestedAppointment
		Appointment	Lexical, designed	X	PossibleAppiontment
		GroupMember	Biddable		SelectPossibleAppointment, SuggestNewAppointment
R03	pdAccess	CC_Access	machine		CheckAccessCalendar ShowAccessCalendarRequest
		MemberPage	connection	X	MakeAccessCalendar ConfirmAccessCalendar
		Group	Lexical, designed		AcessCalendarChecked
		GroupMember	Biddable		RequestAccessCalendar
R04	pdFinalize	CC_Finalize	machine		RemovePreliminaryAppointment, FinalizeAppointment CheckParticipation
		Appointment	Lexical, designed	x	FinalizedAppointment

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

Requirement	**Covered in**	**Contained domain**	**domain type**	**constrained**	**controlled phenomena**
R01	pdAdd	CC_Add	machine		SetNewAppointment, ShowChoosedAppiontmentRequeast
		MemberPage	connection	X	CreatNewAppointment, ConfirmMakingAppointment
		Appointment	Lexical, designed	X	NewAppointment
		GroupMember	Biddable		RequeastNewAppointment
R02	pdAnswer	CC_Answer	machine		SetPossibleAppointment, SetSuggestedAppointment ShowChoosedPossibleAppointment
		MemberPage	connection	X	ConfirmPossibleAppointment CreatPossibleAppointment, CreatSuggestedAppointment
		Appointment	Lexical, designed	X	PossibleAppiontment
		GroupMember	Biddable		SelectPossibleAppointment, SuggestNewAppointment
R03	pdAccess	CC_Access	machine		CheckAccessCalendar ShowAccessCalendarRequest
		MemberPage	connection	X	MakeAccessCalendar ConfirmAccessCalendar
		Group	Lexical, designed		AcessCalendarChecked
		GroupMember	Biddable		RequestAccessCalendar
R04	pdFinalize	CC_Finalize	machine		RemovePreliminaryAppointment, FinalizeAppointment CheckParticipation
		Appointment	Lexical, designed	x	FinalizedAppointment

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

connection domain	phenomenon controlled by connection domain	connected domain	phenomenon controlled by connected Domain
MemberPage	CreatNewAppointment,	GroupMember	RequeastNewAppointment
	ConfirmMakingAppointment	CC_Add	ShowChoosedAppiontmentRequeast
	CreatPossibleAppointment	GroupMember	SelectPossibleAppointment
	CreatSuggestedAppointment	GroupMember	SuggestNewAppointment
	ConfirmPossibleAppointment	CC_Answer	ShowChoosedPossibleAppointment
	MakeAccessCalendar	GroupMember	RequestAccessCalendar
	ConfirmAccessCalendar	CC_Access	ShowAccessCalendarRequest

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

connection domain	phenomenon controlled by connection domain	connected domain	phenomenon controlled by connected Domain
MemberPage	CreatNewAppointment,	GroupMember	RequeastNewAppointment
	ConfirmMakingAppointment	CC_Add	ShowChoosedAppiontmentRequeast
	CreatPossibleAppointment	GroupMember	SelectPossibleAppointment
	CreatSuggestedAppointment	GroupMember	SuggestNewAppointment
	ConfirmPossibleAppointment	CC_Answer	ShowChoosedPossibleAppointment
	MakeAccessCalendar	GroupMember	RequestAccessCalendar
	ConfirmAccessCalendar	CC_Access	ShowAccessCalendarRequest

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

connection domain	phenomenon observed by connection domain	phenomenon controlled by connection domain
MemberPage	RequeastNewAppointment	CreatNewAppointment,
	ShowChoosedAppiontmentRequeast	ConfirmMakingAppointment
	SelectPossibleAppointment	CreatPossibleAppointment
	SuggestNewAppointment	CreatSuggestedAppointment
	ShowChoosedPossibleAppointment	ConfirmPossibleAppointment
	RequestAccessCalendar	MakeAccessCalendar
	ShowAccessCalendarRequest	ConfirmAccessCalendar

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

Provided mapping diagrams.

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

problem diagram	operator	related domains or phenomena
pdAdd	leave out domain	Group, User, Person, Administrator
	introduce connection/display domain	MemberPage
	split interface	CC!{ShowNewAppointment}
	concretize interface	CC!{ShowNewAppointment},GM!{RequeastNewAppointment,RequeastAccessCalendar ,Participate,SelectPossibleAppointment,SuggestNewAppointment}
pdAnswer	leave out domain	Group, User, Person, Administrator
	introduce connection/display domain	MemberPage
	split interface	CC!{ShowPossibleAppointment},GM!{RequeastNewAppointment,RequeastAccessCalendar, Participate, SelectPossibleAppointment,SuggestNewAppointment}
	concretize interface	CC!{ShowNewAppointment}, GM!{RequeastNewAppointment,RequeastAccessCalendar,Participate, SelectPossibleAppointment,SuggestNewAppointment}, AP!{NewAppointment,FinalizedAppointment}
pdAccess	leave out domain	Appointment, User, Person, Administrator
	introduce connection/display domain	MemberPage
	split interface	CC!{AllowAccessCalendar}
	concretize interface	CC!{AllowAccessCalendar}, GM!{RequeastNewAppointment, RequeastAccessCalendar,Participate,SelectPossibleAppointment, SuggestNewAppointment}
pdFinalize	leave out domain	User, Person, Administrator, GroupMember
	introduce connection/display domain	--
	split interface	--
	concretize interface	--

Subproblem for requirement(s):
(R01) fits to **update 2**

(R02) fits to **update 2**

(R03) fits to **query 2**

(R04) fits to **simple transformation**

The Validation:

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

Problem Diagram	Problem Frame	Connections in PD	Connections in PF	Domain Type 1	Domain Type 2
Add	Update2	CCA![SetNewAppointment} AP!{NewAppointment}	DB!Y1,UM!E2	Machine	LexicalDomain
		CCA!{ShowChoosed AppiontmentRequeast} MP!{CreatNewAppointment}	UM!E4,IOD!E8	Machine	ConnectionDomain
		GM!{RequeastNewAppointment} MP![ConfirmMakingAppointment}	UO!E6,IO!C7	BiddableDomain	ConnectionDomain
Answer	Update2	AP!{PossibleAppiontment} CCAN!{SetPossibleAppointment, SetSuggestedAppointment}	DB!Y1,UM!E2	Machine	LexicalDomain
		MP!{CreatPossibleAppointment, CreatSuggestedAppointment} CCAN!{ShowChoosed PossibleAppointment}	UM!E4,IOD!E8	Machine	ConnectionDomain
		GM!{SelectPossibleAppointment, SuggestNewAppointment} MP!{ConfirmPossibleAppointment}	UO!E6,IO!C7	BiddableDomain	ConnectionDomain
Access	Query2	CCAC!{CheckAccessCalendar} G!{AcessCalendarChecked}	DB!Y1	Machine	LexicalDomain
		CCAC!{ShowAccessCalendarRequest} MP!{MakeAccessCalendar}	QM!Y3,IOD!C6	Machine	ConnectionDomain
		GM!{RequestAccessCalendar} MP!{ConfirmAccessCalendar}	IOD!E7,EO!E5	BiddableDomain	ConnectionDomain

Finalize	Simple tranformation	CCF![RemovePreliminaryAppointment,FinalizeAppointment} AP!{FinalizedAppointment}	I!Y!	Machine	LexicalDomain
Problem Diagram	Problem Frame	Connections in PD	Connections in PF	Domain Type 1	Domain Type 2
Add	Update2	CCA![SetNewAppointment} AP!{NewAppointment}	DB!Y1,UM!E2	Machine	LexicalDomain
		CCA!{ShowChoosed AppiontmentRequeast} MP!{CreatNewAppointment}	UM!E4,IOD!E8	Machine	ConnectionDomain
		GM!{RequeastNewAppointment} MP![ConfirmMakingAppointment}	UO!E6,IO!C7	BiddableDomain	ConnectionDomain
Answer	Update2	AP!{PossibleAppiontment} CCAN!{SetPossibleAppointment, SetSuggestedAppointment}	DB!Y1,UM!E2	Machine	LexicalDomain
		MP!{CreatPossibleAppointment, CreatSuggestedAppointment} CCAN!{ShowChoosed PossibleAppointment}	UM!E4,IOD!E8	Machine	ConnectionDomain

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

Problem Diagram	**Problem Frame**	**Constrained Domains in PD**	**Constrained Domains in PF**	**Domain Type**
Add	Update2	Appointment	Data Base	LexicalDomain
		MemberPage	Input Output Device	ConnectionDomain
Answer	Update2	Appointment	Data Base	LexicalDomain
		MemberPage	Input Output Device	ConnectionDomain
Access	Query2	MemberPage	Input Output Device	ConnectionDomain
Finalize	transformation	Appointment	Outputs	LexicalDomain

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

Problem Diagram	**Problem Frame**	**Referred Domains in PD**	**Referred Domains in PF**	**Domain Type**
Add	Update2	GroupMember	Update Operator	BiddableDomain
Answer	Update2	GroupMember	Update Operator	BiddableDomain
Access	Query2	Group	Data Base	LexicalDomain
		GroupMember	Enquiry Operator	BiddableDomain

A3

R01: Add Appointment (R4+R5)

Every group member can create a new appointment request in the group's calendar.
All dates at which the appointment may take place shall be added to the group calendar and marked as preliminary appointment dates.

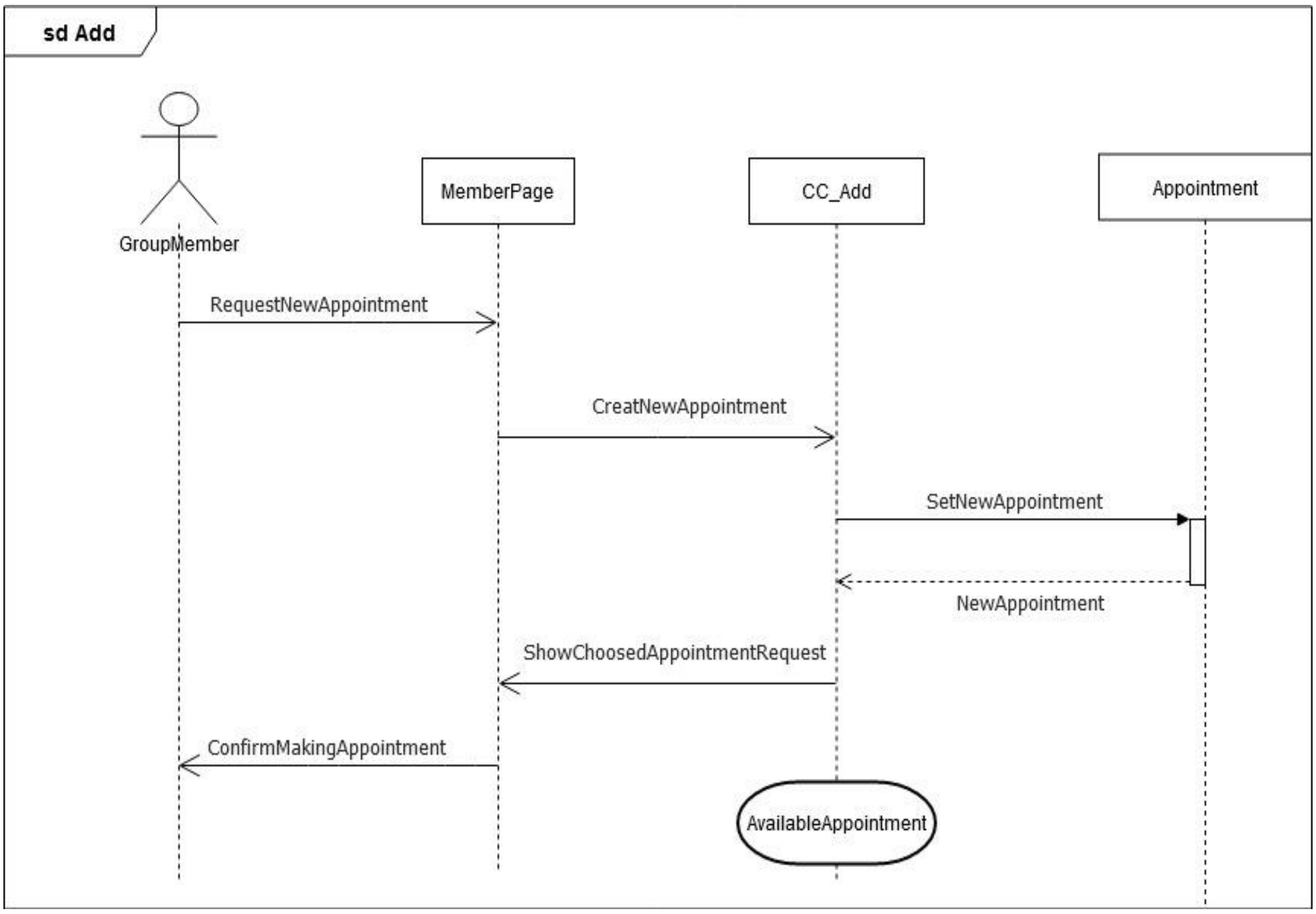
MemberPage (S01a): When the MemberPage receives the command "RequestNewAppointment", then the command is forwarded to the machine with the command "CreatNewAppointment". The results are received via the command "ShowChoosedAppointmentRequest"and shown to the GroupMember by "ConfirmMakingAppointment".

CC_Add (S01b): When the machine receives the command "CreatNewAppointment", the available new Appointment is selected with the command "SetNewAppointment" received as the data "NewAppointment". The results are returned via the command "ShowChoosedAppointmentRequest".

Appointment (S01c): After receiving the command "SetNewAppointment" the new appointment is returned as the data "NewAppointment".

Correctness condition: (S01a) ∧ (S01b) ∧ (S01c) ∧ (R4) ∧ (R5) ⇒ (R01)

sequence diagram (Add):



R02: Answer Appointment(R8)

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.

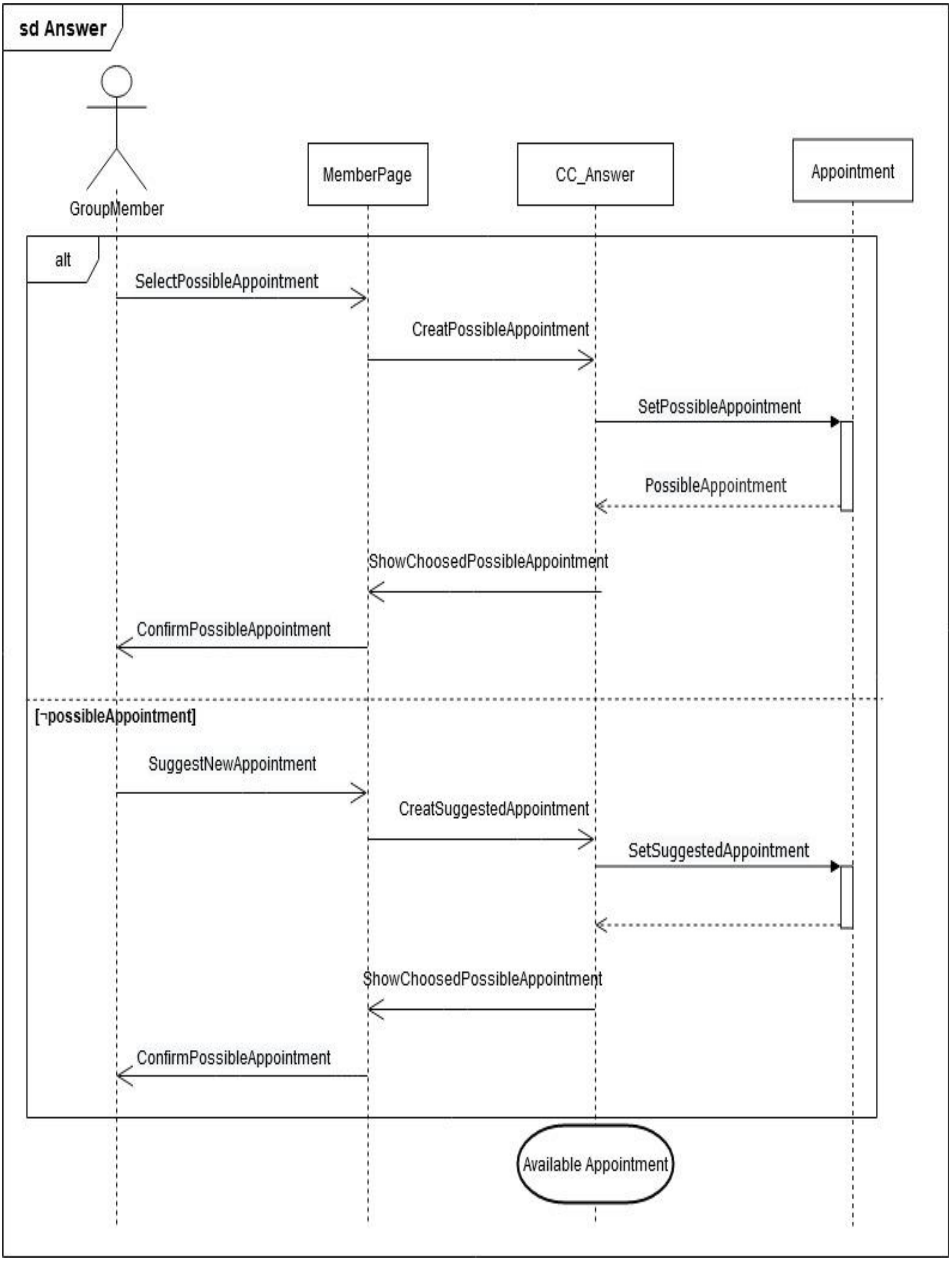
MemberPage (S02a) When the MemberPage receives the commands "SelectPossibleAppointment,SuggestNewAppointment", then the command is forwarded to the machine with the command "CreatPossibleAppointment,CreatSuggestedAppointment". The results are received via the command "ShowChoosedPossibleAppointment "and shown to the GroupMember by "ConfirmPossibleAppointment".

CC_Answer (S02b): When the machine receives the command "CreatPossibleAppointment", the available possible Appointment is selected with the command "SetPossibleAppointment" and received as the data "PossibleAppointment". And if the suggested appointments are available then the machine sets them with the command "SetSuggestedAppointment". The results are returned via the command "ShowChoosedPossibleAppointment".

Appointment (S02c): After receiving the command "SetPossibleAppointment" the possible Appointment is returned as the data "PossibleAppointment". When the commands "SetSuggestedAppointment" are received, the suggested Appointment is set.

Correctness condition: (S02a) ∧ (S02b) ∧ (S02c) ∧ (R8) ∧ (A4) ∧ (A5) ∧ (A6) ∧ (A7) ∧ (A8)⇒ (R02)

sequence diagram (Answer):



R03: Show Calendar(R9)

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

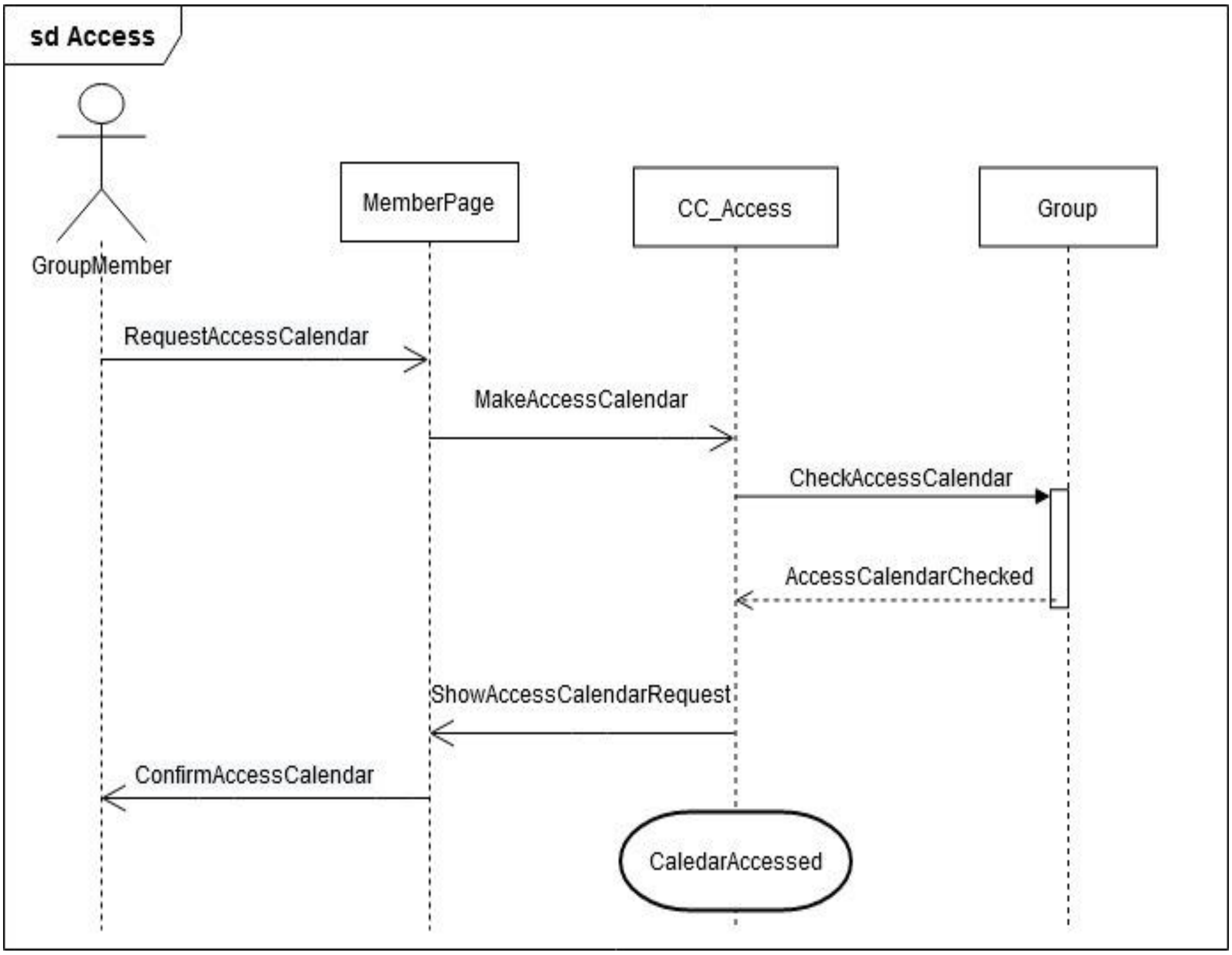
MemberPage (S03a): When the MemberPage receives the command "RequestAccessCalendar", then the command is forwarded to the machine with the command "MakeAccessCalendar". The results are received via the command "ShowAccessCalendarRequest "and shown to the GroupMember by "ConfirmAccessCalendar".

CC_Access (S03b): When the machine receives the command "MakeAccessCalendar", the available access calendar request is selected with the command "CheckAccessCalendar" and received as the data "AcessCalendarChecked". The results are returned via the command "ShowAccessCalendarRequest".

Group (S03c): After receiving the commands "CheckAccessCalendar" the result is returned as the data "AcessCalendarChecked".

Correctness condition: (S03a) ∧ (S03b) ∧ (S03c) ⇒ (R03)

sequence diagram (Access):



R04: Finalize Appointment (R5+R7)

All dates at which the appointment may take place shall be added to the group calendar and marked as preliminary appointment dates.

The chosen date is added as fixed date of the appointment to the calendar and all other preliminary dates of the appointment are removed from it. Additionally, all planned participants for whom the date is possible are recorded as actual participant.

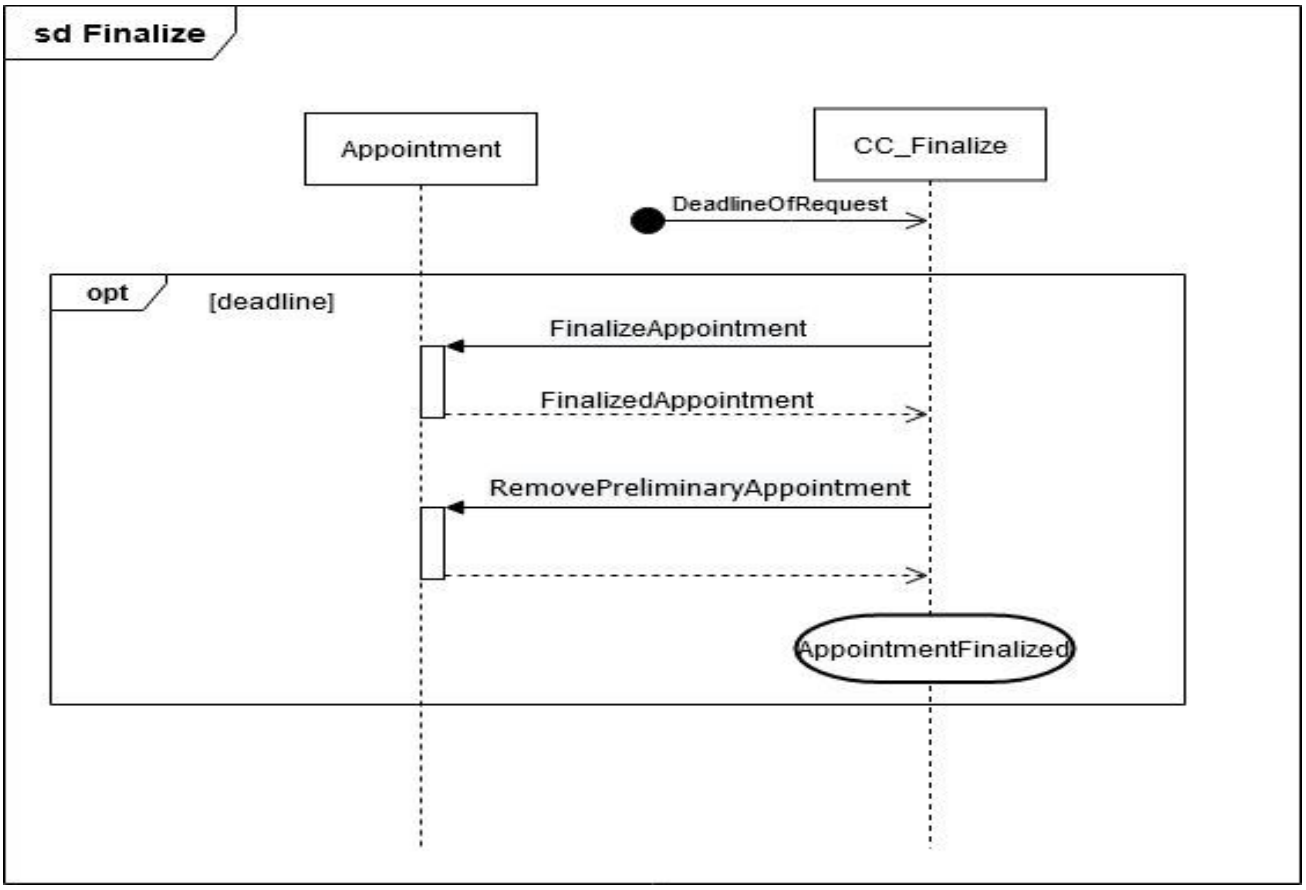
Appointment (S04a): When receiving “FinalizeAppointment” the fix appointment is marked.

CC_Finalize (S04b): When the machine receives the command “DeadlineOfRequest”, the participation is checked using the command “CheckParticipation” and received as the Data “ParticipationChecked”. The fix Appointment is set using the command “FinalizeAppointment”. All preliminary appointments are removed using the command “RemovePreliminaryAppointment”. DeadlineOfRequest is a sending event and the sender is not known and that can be triggered by timer.

GroupMember (A4) All group members check regularly whether new appointment requests exist they
shall participate in.

Correctness condition: (S04a) ∧ (S04b) ∧ (S04c) ∧ (F2) ⇒ (R04)

sequence diagram (Finalize):



The Validation:

- $S_{abstract} \wedge D$ are non-contradictory.

  No contradictions can be found in $S_{abstract} \wedge D$

- $S_{abstract} \wedge D \Longrightarrow R$.

  (S01a) ∧ (S01b) ∧ (S01c) ∧ (R4) ∧ (R5) ⇒ (R01)

  (S02a) ∧ (S02b) ∧ (S02c) ∧ (R8) ∧ (A4) ∧ (A5) ∧ (A6) ∧ (A7) ∧ (A8) ⇒ (R02)

  (S03a) ∧ (S03b) ∧ (S03c) ⇒ (R03)

  (S04a) ∧ (S04b) ∧ (S04c) ∧ (F2) ⇒ (R04)

- Messages and phenomena are consistent

message in scenario	**Source**	**target**	**phenomena in problem diagram**
RequestNewAppointment	GroupMember	MemberPage	GM!{RequestNewAppointment}
CreatNewAppointment	MemberPage	CC_Add	MP!{CreatNewAppointment}
SetNewAppointment	CC_Add	Appiontment	AP!{NewAppointment}
ShowChoosedAppointmentRequest	CC_Add	MemberPage	CCA!{ShowChoosedAppointmentRequest }
ConfirmMakingAppointment	MemberPage	GroupMember	MP!{ ConfirmMakingAppointment }
SelectPossibleAppointment	GroupMember	MemberPage	GM!{ SelectPossibleAppointment }
CreatPossibleAppointment	MemberPage	CC_Answer	MP!{ CreatPossibleAppointment }
SetPossibleAppointment	CC_Answer	Appointment	AP!{PossibleAppointment}
SuggestNewAppointment	GroupMember	MemberPage	GM!{ SuggestNewAppointment }
CreatSuggestedAppointment	MemberPage	CC_Answer	MP!{ CreatSuggestedAppointment }
SetSuggestedAppointment	CC_Answer	Appointment	CCAN!{SetSuggestedAppointment}
ShowChoosedPossibleAppointment	CC_Answer	MemberPage	CCAN!{ShowChoosedPossibleAppointment}
ConfirmPossibleAppointment	MemberPage	GroupMember	MP!{ConfirmPossibleAppointment}
RequestAccessCalendar	GroupMember	MemberPage	GM!{ RequestAccessCalendar }
MakeAccessCalendar	MemberPage	CC_Access	MP!{ MakeAccessCalendar }
AccessCalendarChecked	Group	CC_Access	G!{ AccessCalendarChecked }
CheckAccessCalendar	CC_Access	Group	CCAC!{ CheckAccessCalendar }
ShowAccessCalendarRequest	CC_Access	MemberPage	CCAC!{ ShowAccessCalendarRequest }
ConfirmAccessCalendar	MemberPage	GroupMember	MP!{ ConfirmAccessCalendar }
DeadlineOfRequest	--	CC_Finalize	timed event
FinalizeAppointment	CC_Finalize	Appointment	CCF!{FinalizeAppointment}
RemovePreliminaryAppointment	CC_Finalize	Appointment	CCF!{RemovePreliminaryAppointment }

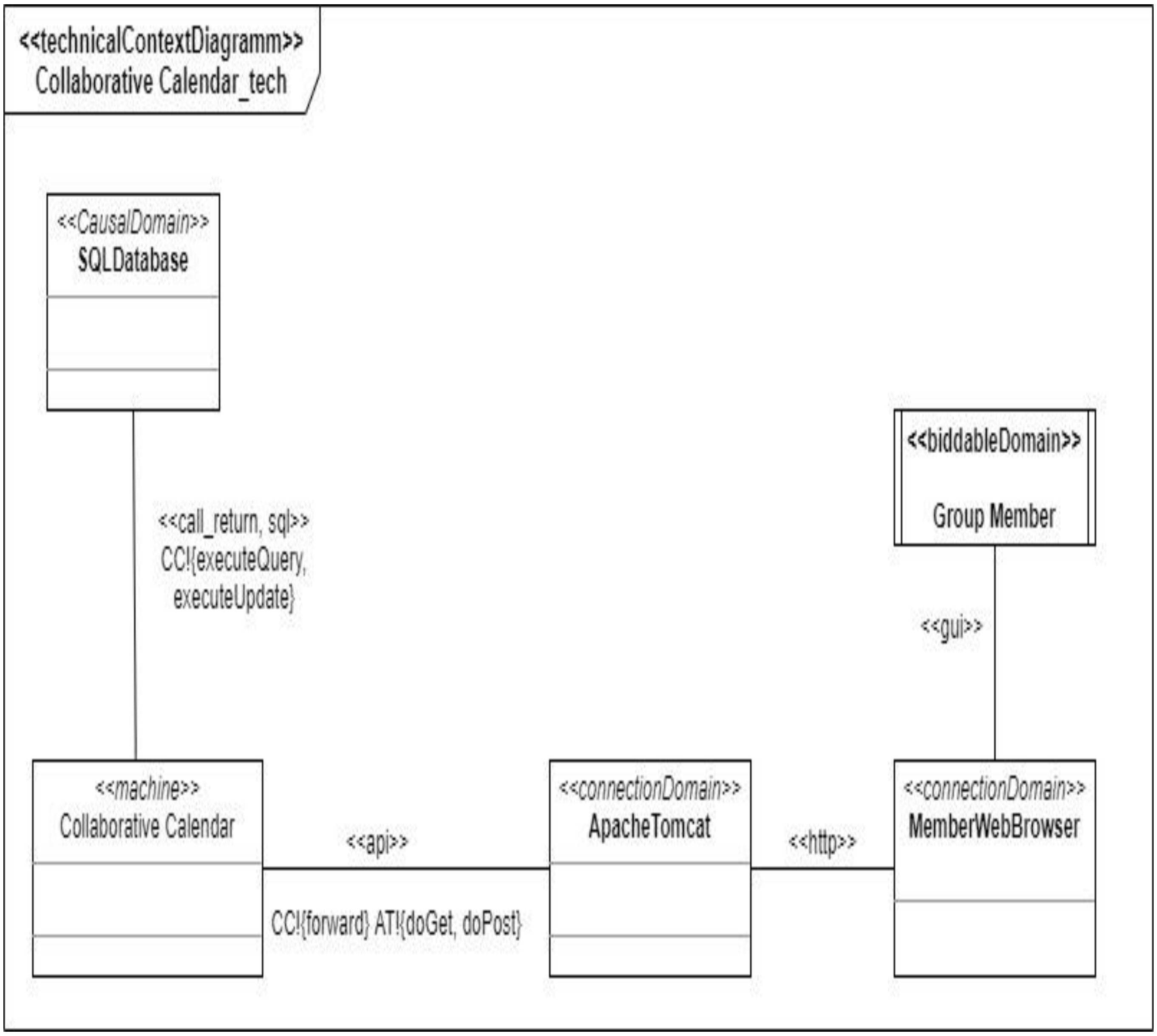
- Lexical domains are not sources of messages

message in scenario	source	domain type
RequestNewAppointment	GroupMember	BiddableDomain
CreatNewAppointment	MemberPage	ConnectionDomain
SetNewAppointment	CC_Add	Machine
ShowChoosedAppointmentRequest	CC_Add	Machine
ConfirmMakingAppointment	MemberPage	ConnectionDomain
SelectPossibleAppointment	GroupMember	BiddableDomain
CreatPossibleAppointment	MemberPage	ConnectionDomain
SetPossibleAppointment	CC_Answer	Machine
SuggestNewAppointment	GroupMember	BiddableDomain
CreatSuggestedAppointment	MemberPage	ConnectionDomain
SetSuggestedAppointment	CC_Answer	Machine
ShowChoosedPossibleAppointment	CC_Answer	Machine
ConfirmPossibleAppointment	MemberPage	ConnectionDomain
RequestAccessCalendar	GroupMember	BiddableDomain
MakeAccessCalendar	MemberPage	ConnectionDomain
AccessCalendarChecked	Group	lexicalDomain
CheckAccessCalendar	CC_Access	Machine
ShowAccessCalendarRequest	CC_Access	Machine
ConfirmAccessCalendar	MemberPage	ConnectionDomain
DeadlineOfRequest	--	--
RemovePreliminaryAppointment	CC_Finalize	Machine
FinalizeAppointment	CC_Finalize	Machine
FinalizedAppointment	Appointment	lexicalDomain

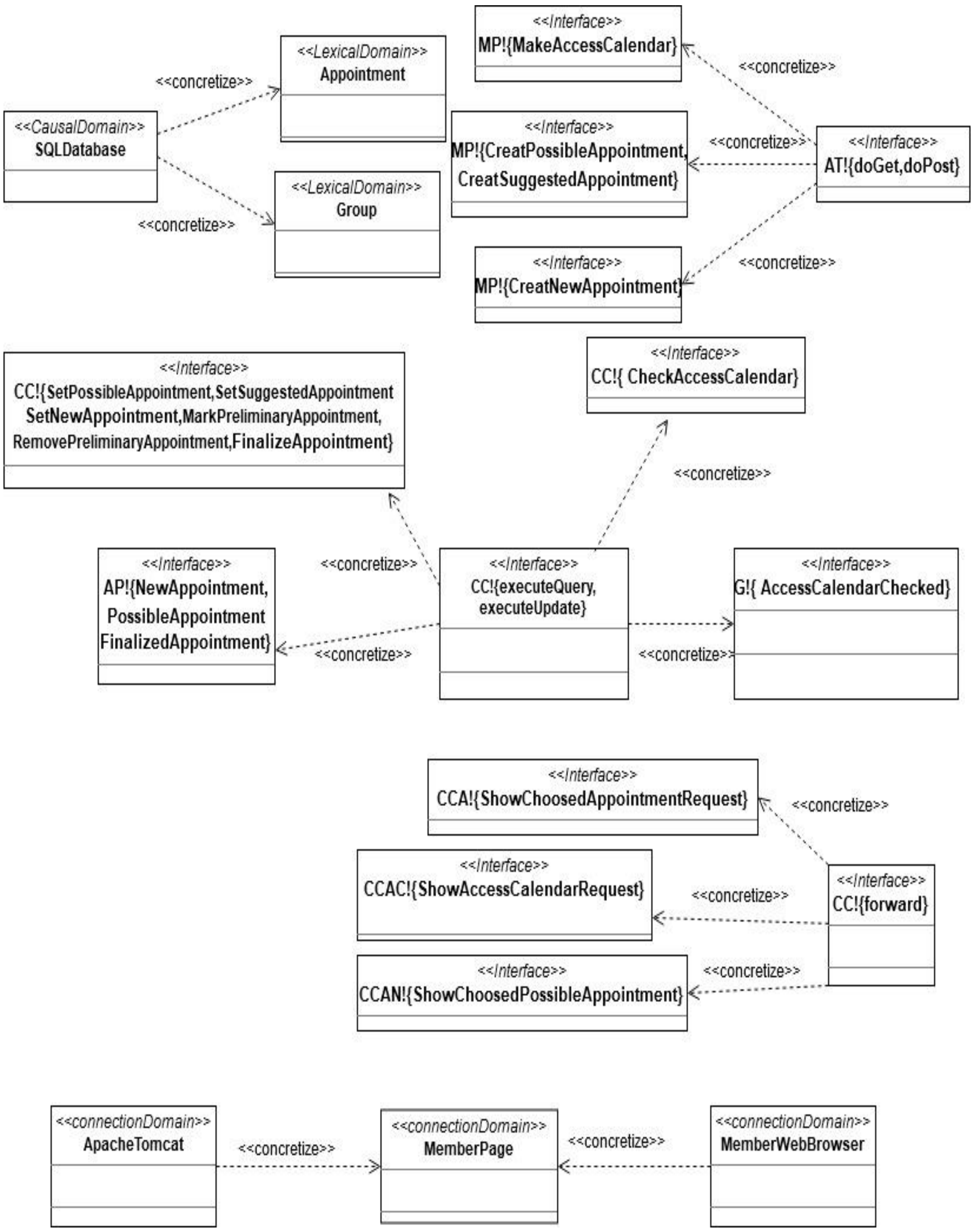
- There exists at least one scenario for each subproblem.
- Scenarios cover normal cases and possibly exceptional cases.

subproblem	normal case	exceptional case
pdAdd	sdAdd	--
pdAnswer	sdAnswer	--
pdAccess	sdAccess	--
pdFinalize	sdFinailize	--

Technical software specification:



Mapping diagram for Technical Context Diagram:


<<CausalDomain>> SQLDatabase
<<concretize>>
<<LexicalDomain>> Appointment
<<concretize>>
<<LexicalDomain>> Group
<<Interface>> MP!{MakeAccessCalendar}
<<concretize>>
<<Interface>> MP!{CreatPossibleAppointment, CreatSuggestedAppointment}
<<concretize>>
<<Interface>> AT!{doGet,doPost}
<<Interface>> MP!{CreatNewAppointment}
<<concretize>>
<<Interface>> CC!{SetPossibleAppointment,SetSuggestedAppointment SetNewAppointment,MarkPreliminaryAppointment, RemovePreliminaryAppointment,FinalizeAppointment}
<<Interface>> CC!{ CheckAccessCalendar}
<<concretize>>
<<Interface>> AP!{NewAppointment, PossibleAppointment FinalizedAppointment}
<<concretize>>
<<Interface>> CC!{executeQuery, executeUpdate}
<<Interface>> G!{ AccessCalendarChecked}
<<concretize>>
<<concretize>>
<<Interface>> CCA!{ShowChoosedAppointmentRequest}
<<concretize>>
<<Interface>> CCAC!{ShowAccessCalendarRequest}
<<concretize>>
<<Interface>> CC!{forward}
<<Interface>> CCAN!{ShowChoosedPossibleAppointment}
<<concretize>>
<<connectionDomain>> ApacheTomcat
<<concretize>>
<<connectionDomain>> MemberPage
<<concretize>>
<<connectionDomain>> MemberWebBrowser

Technical software specification - Validation:

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

Message	**new phenomena and domains**
RequestNewAppointment	ApacheTomcat, HTTP
CreatNewAppointment	ApacheTomcat, HTTP
SelectPossibleAppointment	ApacheTomcat, HTTP
SelectSuggestedAppointment	ApacheTomcat, HTTP
CreatPossibleAppointment	ApacheTomcat, HTTP
CreatSuggestedAppointment	ApacheTomcat, HTTP
RequestAccessCalendar	ApacheTomcat, HTTP
MakeAccessCalendar	ApacheTomcat, HTTP

All internal messages can be realized using SQL commands.

- All domains of the technical context diagram are related to domains in the problem diagrams.
- All phenomena in the technical context diagram are related to elements in the problem diagrams.

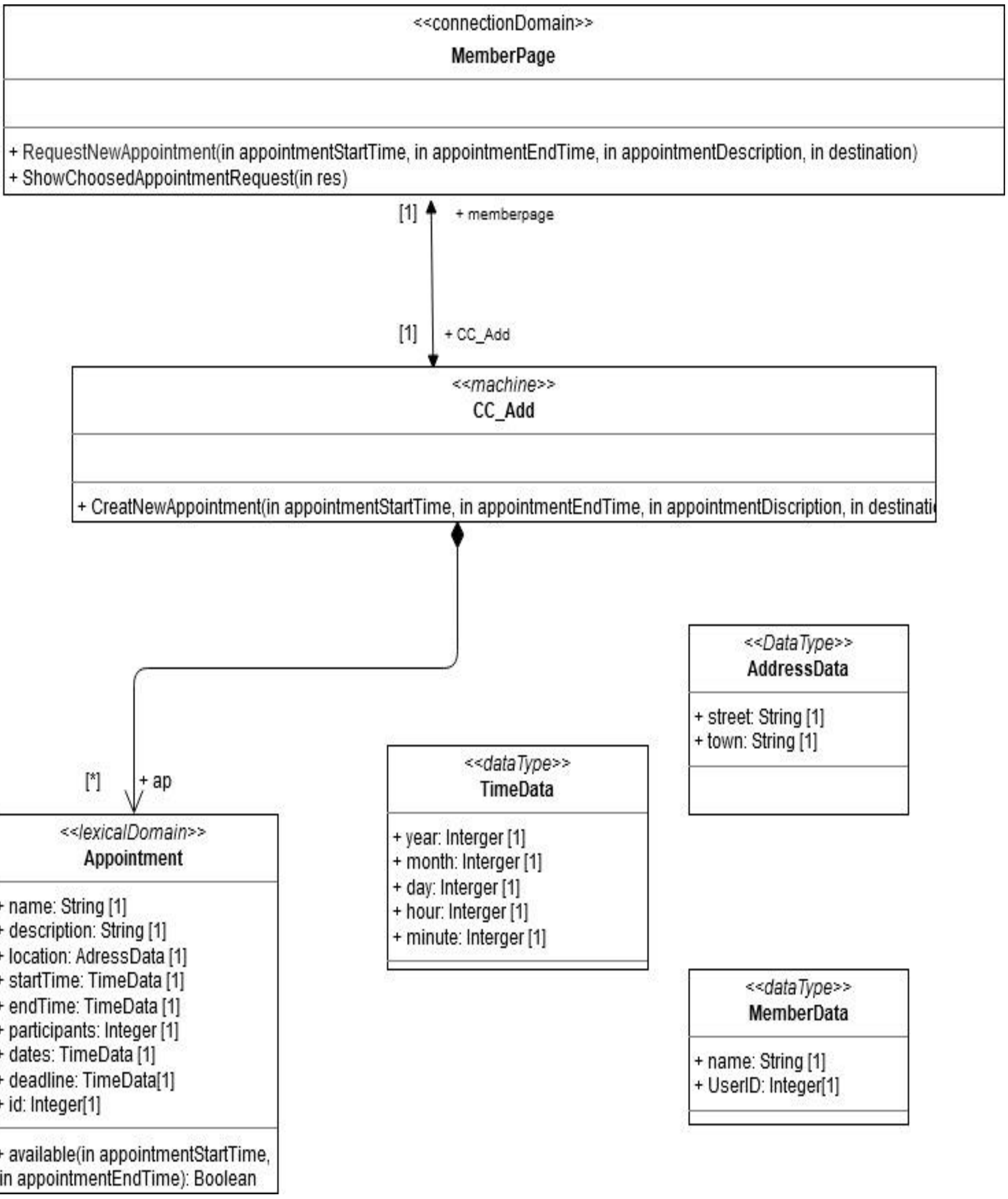
Provided mapping diagram.

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

Problem Diagram	**Domain connected with the machine**	**Element in the TCD**
Add	MemberPage	MemberWebBrowser, ApacheTomcat
	Appointment	SQLDatabase
Answer	MemberPage	MemberWebBrowser, ApacheTomcat
	Appointment	SQLDatabase
Access	MemberPage	MemberWebBrowser, ApacheTomcat
	Group	SQLDatabase
Finalize	Appointment	SQLDatabase

The operation CreatNewAppointment

Class model:



The operation RequestNewAppointment

Operation specification:

Name: RequestNewAppointment

Description: Forwards new appointment request from the GroupMember to the machine.

OCL constraint:

```
context MemberPage :: RequestNewAppointment( appointmentStartTime : TimeData,
    appointmentEndTime : TimeData, appointmentDescription : String, destination:AddressData)
Pre: true
Post: CC_Add^CreatNewAppointment ( appointmentStartTime,  appointmentEndTime,
    appointmentDescription, destination)
```

The operation CreatNewAppointment

Operation specification

Name: CreatNewAppointment

Description: Generates and return list of appointments matching the input criteria concerning appointment startTime, appointment endTime, appointmentDescription and destination

OCL constraint:

```
context CC_Add :: CreatNewAppointment( appointmentStartTime : TimeData,
    appointmentEndTime : TimeData, appointmentDescription : String, destination:AddressData)
Pre: true
Post: let res: Set (Appointment) = ap ->one (a: Appointment |
            a.startTime = appointmentStartTime and
            a.endTime = appoimtentEndTime  and
            a.available (appointmentStartTime, appoimtentEndTime )) ->asSet()
        in
            MemberPage^ShowChoosedAppointmentRequest( res )
```

The operation CreatNewAppointment

Invariant

Every appointment has to be identified by a unique id

OCL constraint

```
context Appointment
inv: Appointment. allInstances()->isUniqe(id)
```

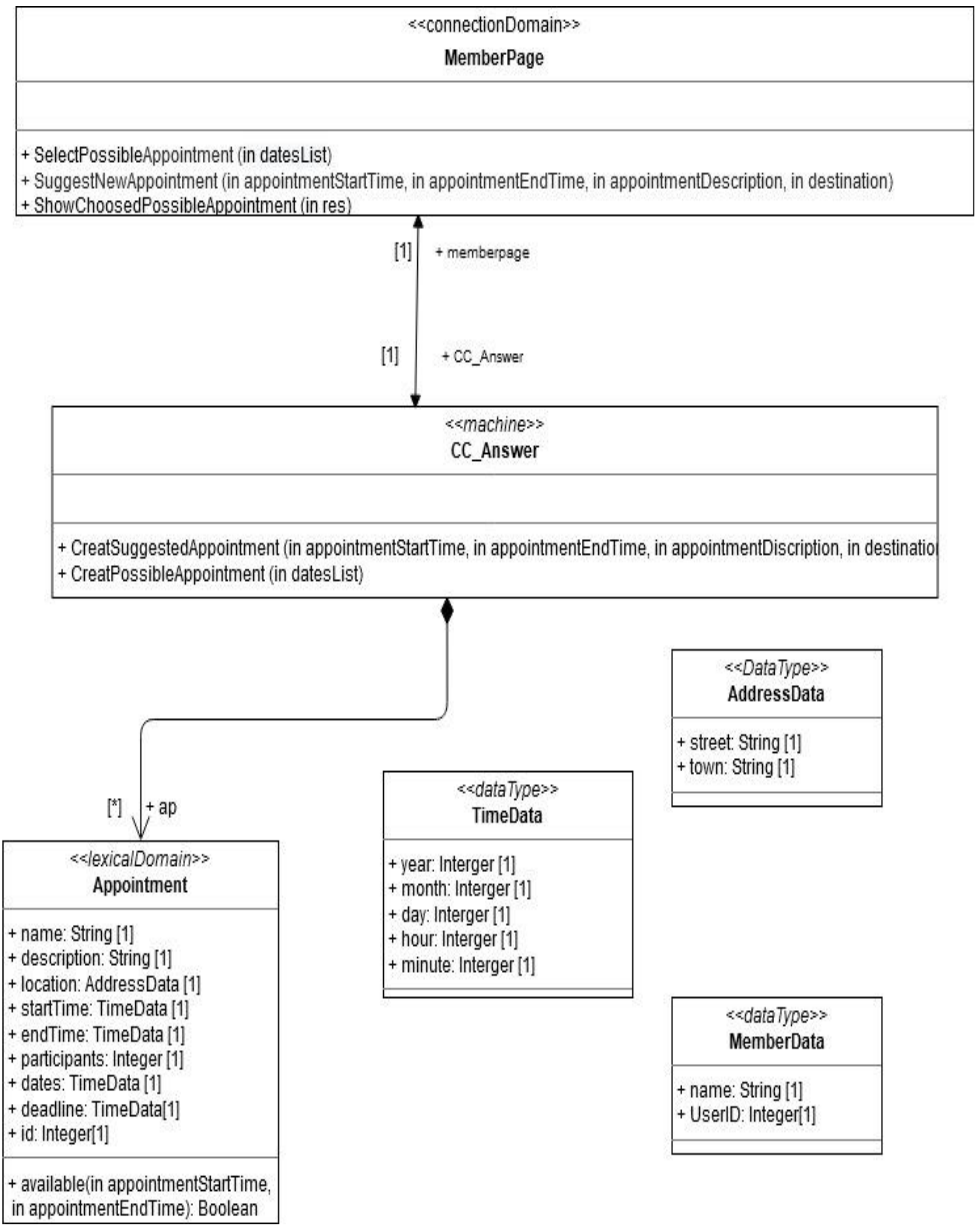
The operation CreatNewAppointment

The validation:

- Operation specifications must be consistent with abstract specifications:
  The operation specification of RequestNewAppointment is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition.
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  GroupMember can input all parameters to MemberPage, which forwards these to this operation.
  The machine knows the argument res used in the message to GroupMember.
- Parameters must be used in the pre- and/or postcondition:
  The parameters are used in the postcondition.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  The method available (appointmentStartTime:TimeData, appointmentEndTime:TimeData):Boolean added to the class Appointment because we need to know what appointments are available.
- The attribute id was introduced for practical reasons.

The operation CreatSuggestedAppointment / CreatPossibleAppointment

Class model:



<<connectionDomain>>
MemberPage
+ SelectPossibleAppointment (in datesList)
+ SuggestNewAppointment (in appointmentStartTime, in appointmentEndTime, in appointmentDescription, in destination)
+ ShowChoosedPossibleAppointment (in res)
[1]
+ memberpage
[1]
+ CC_Answer
<<machine>>
CC_Answer
+ CreatSuggestedAppointment (in appointmentStartTime, in appointmentEndTime, in appointmentDiscription, in destinatio
+ CreatPossibleAppointment (in datesList)
[*]
+ ap
<<lexicalDomain>>
Appointment
+ name: String [1]
+ description: String [1]
+ location: AddressData [1]
+ startTime: TimeData [1]
+ endTime: TimeData [1]
+ participants: Integer [1]
+ dates: TimeData [1]
+ deadline: TimeData[1]
+ id: Integer[1]
+ available(in appointmentStartTime, in appointmentEndTime): Boolean
<<dataType>>
TimeData
+ year: Interger [1]
+ month: Interger [1]
+ day: Interger [1]
+ hour: Interger [1]
+ minute: Interger [1]
<<DataType>>
AddressData
+ street: String [1]
+ town: String [1]
<<dataType>>
MemberData
+ name: String [1]
+ UserID: Integer[1]

The operation SuggestNewAppointment / SelectPossibleAppointment

Operation specification

Name: SuggestNewAppointment / SelectPossibleAppointment

Description: Forwards new suggested appointment request from the GroupMember to the machine. / Forwards the possible Appointment from the GroupMember to the machine.

OCL constraint:

```
context MemberPage :: SuggestNewAppointment ( appointmentStartTime: TimeData,
appointmentEndTime: TimeData, appointmentDescription: String, destination: AddressData)
    Pre: true
Post: CC_Answer ^ CreatSuggestedAppointment (appointmentStartTime, appointmentEndTime,
appointmentDiscription, destination)
```

OCL constraint:

```
context MemberPage :: SelectPossibleAppointment (datesList: TimeData).
    Pre: true
Post: CC_Answer ^ CreatPossibleAppointment(datesList)
```

The operation CreatSuggestedAppointment / CreatPossibleAppointment

Operation specification:

Name: CreatSuggestedAppointment / CreatPossibleAppointment

Description: Generates and return list of suggested appointments matching the input criteria concerning appointment startTime, appointment endTime, appointmentDescription and destination. / Generates and return list of possible appointments matching the input criteria concerning possible appointments list.

OCL constraint:

```
context CC_Answer :: CreatPossibleAppointment (datesList: TimeData).
Pre: true
Post: let res: Set (Appointment) = ap -> select (a: Appointment |
              a.id = aid and
               a.available (appointmentStartTime, appoimtentEndTime )) ->asSet() and
        in
               MemberPage ^ ShowChoosedPossibleAppointment(res)
```

```
Context CC_Answer :: CreatSuggestedAppointment(appointmentStartTime : TimeData,
    appointmentEndTime : TimeData, appointmentDescription : String, destination :
    AddressData).
Pre: true
Post: let res: Set (Appointment) = ap ->one (b: Appointment |
            b.startTime = appointmentStartTime and
            b.endTime = appoimtentEndTime and
            b.available (appointmentStartTime, appoimtentEndTime )) ->asSet() and
```

The operation CreatSuggestedAppointment / CreatPossibleAppointment

Invariant

Every appointment has to be identified by a unique id.

OCL constraint:

```
context Appointment
inv: Appointment. allInstances()->isUniqe(id)
```

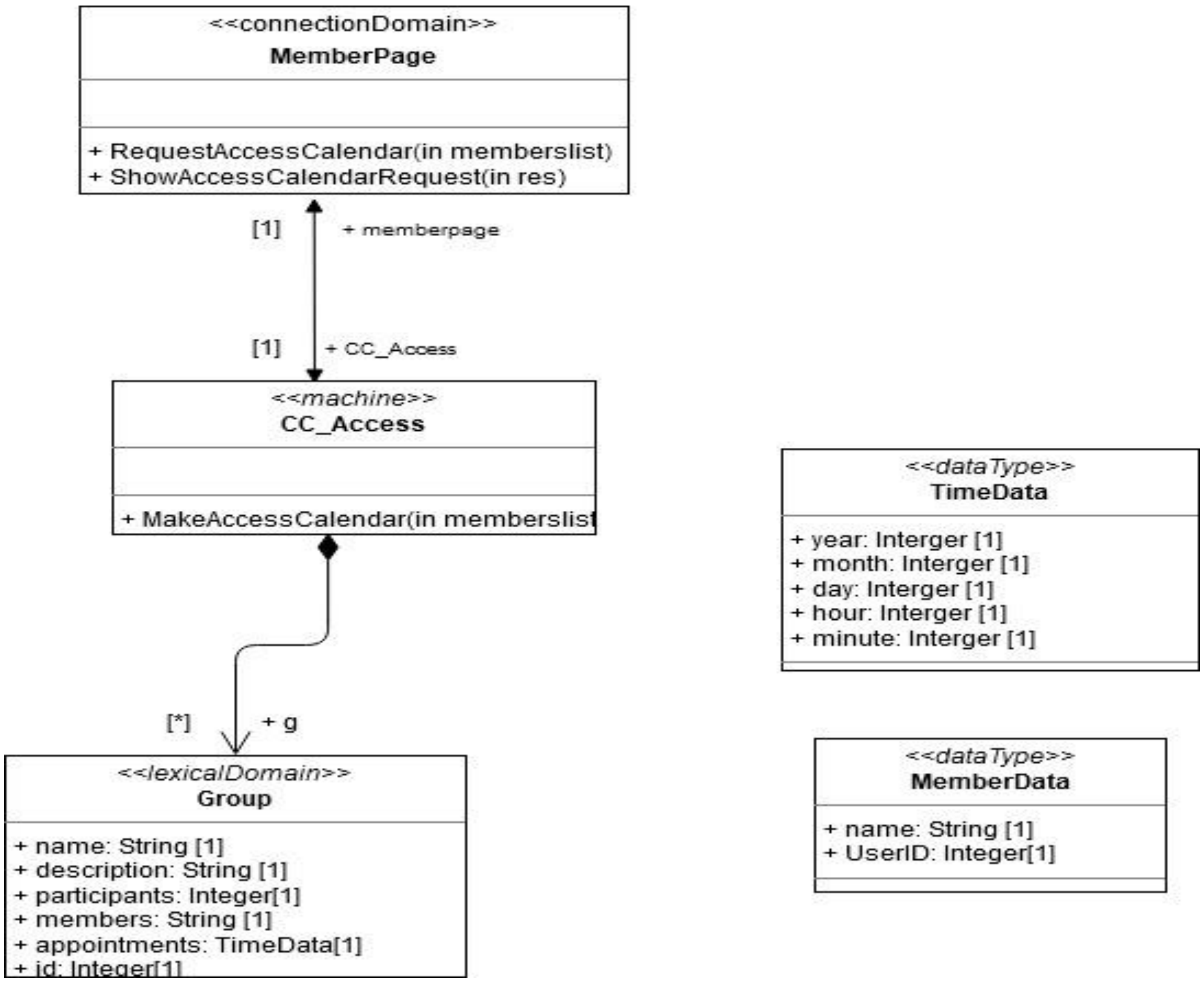
The operation CreatSuggestedAppointment / CreatPossibleAppointment

The validation:

- Operation specifications must be consistent with abstract specifications:
  The operation specification of SelectPossibleAppointment is consistent with the abstract specification.
  The operation specification of SudggestNewAppointment is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition.
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  GroupMember can input all parameters to MemberPage, which forwards these to this operation.
  The machine knows the argument res used in the message to GroupMember.
- Parameters must be used in the pre- and/or postcondition:
  The parameters are used in the postcondition.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  The method available (appointmentStartTime:TimeData, appointmentEndTime:TimeData):Boolean is added to the class Appointment because we need to know what appointments are available.
- The attribute id was introduced for practical reasons.

The operation RequestAccessCalendar

Class model:



The operation RequestAccessCalendar

Operation specification:

Name: RequestAccessCalendar

Description: Forwards new access Calendar request from the GroupMember to the machine.

OCL constraint:

```
context MemberPage :: RequestAccessCalendar (memberslist: String)
Pre: true
Post: CC_Access^MakeAccessCalendar (memberslist)
```

The operation MakeAccessCalendar

Operation specification

Name: MakeAccessCalendar

Description MakeAccessCalendar**:** Generates and return list of members matching the input criteria concerning memberslist.

OCL constraint:

```
context CC_Access :: MakeAccessCalendar (memberslist: String)
Pre: true
Post: let res: Set (Group) = g ->select (a: group | a.members = memberslist) ->asSet()
        in
            MemberPage^ShowAccessCalendarRequest(res)
```

The operation MakeAccessCalendar

Invariant

Every group has to be identified by a unique id

OCL constraint

```
context Group
inv: Group. allInstances()->isUniqe(id)
```

The operation MakeAccessCalendar

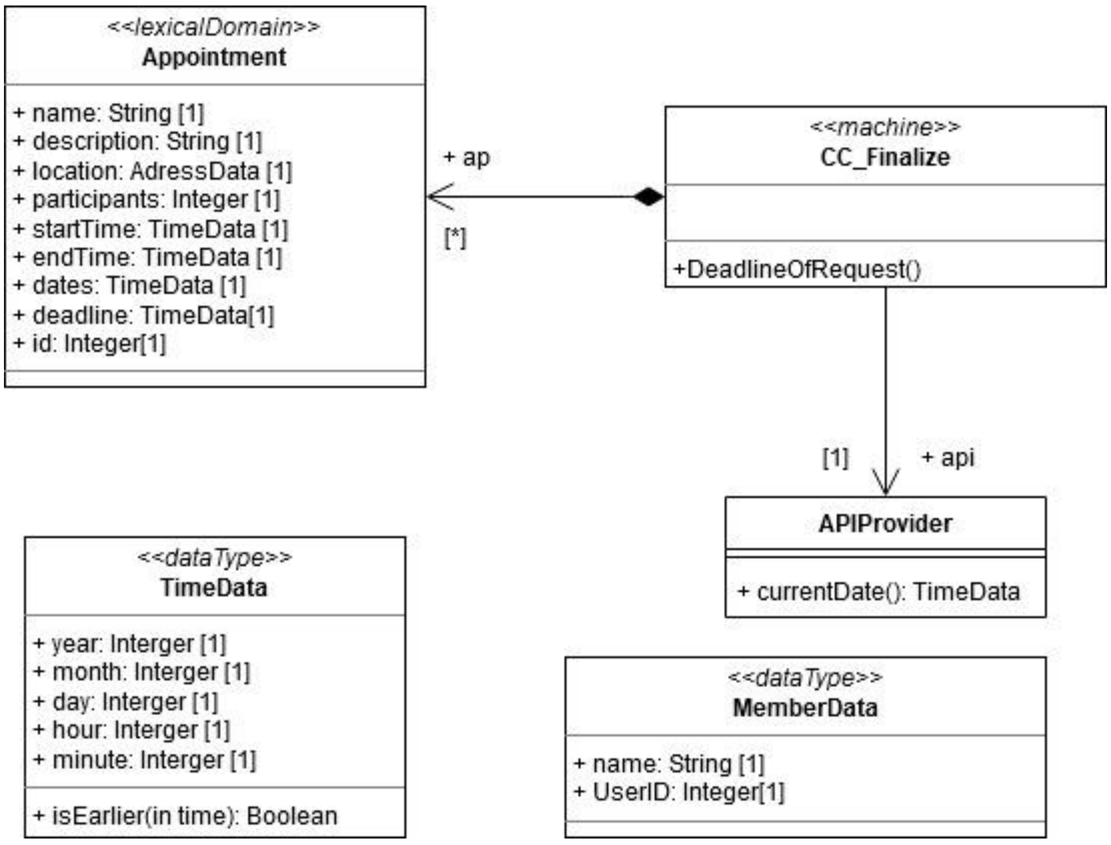
The validation:

- Operation specifications must be consistent with abstract specifications:
  The operation specification of RequestAccessCalendar is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition.
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  GroupMember can input all parameters to MemberPage, which forwards these to this operation.
  The machine knows the argument res used in the message to GroupMember.
- Parameters must be used in the pre- and/or postcondition:
  The parameters are used in the postcondition.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification.

The attribute id was introduced for practical reasons.

The operation DeadlineOfRequest

Class model:



The operation DeadlineOfRequest

Operation specification

Name: DeadlineOfRequest

Description: this internal operation checks whether the request of the new Appointment has reached its deadline and the Appointment will be finalized after the Participants have been checked and the Deadline is reached.

OCL constraint:

```
context CC_Finalize :: DeadlineOfRequest ()
Pre: true
Post: let  PreliminaryAppointmentToBeDeleted: set (Appointment) = ap@pre ->select (a:
    Appointment | a.deadline.isEarlier(currentDate()) and a.participants = Null) ->asSet()
       in
           ap=ap@pre ->-(PreliminaryAppointmentToBeDeleted)
```

The operation DeadlineOfRequest

Invariant

Every appointment has to be identified by a unique id

OCL constraint

```
context Appointment
inv: Appointment. allInstances()->isUniqe(id)
```

The operation DeadlineOfRequest

The validation:

- Operation specifications must be consistent with abstract specifications:
  The operation specification of DeadlineOfRequest is consistent with the abstract specification.
- The postcondition covers all cases exhibited in the abstract specification:
  The normal case behavior described in the abstract specification is covered in the postcondition.
- All parameters of operations must be known by the caller and all parameters of sent messages must be known by the machine:
  The Operation has no parameters.
- Parameters must be used in the pre- and/or postcondition:
  The Operation has no parameters.
- All classes, associations, and attributes newly introduced in the class model must be motivated by some operation specification:
  The Method isEarlier(time: TimeData): Boolean are added to the data type TimeData because we need to compare modified dates.

The attribute id was introduced for practical reasons.

State predicate definitions:

State predicate	Parameters	Definition
AvailableAppointment	aid	ap->one (a:Appointment| a.id = aid)
AvailablePossibleAppointment	aid	ap->exists (a:Appointment| a.id = aid)
AvailableSuggestedAppointment	aid	ap->one (a:Appointment| a.id = aid)
AppointmentFinalized	aid	ap->one (a:Appointment| a.id = aid and a.participants not Null)

Collaborative Calendar life-cycle:

$LC_{\text{group member}} = (Add \mid Access \mid Answer)^*$

$LC_{\text{collaborative calendar}} = (||^{n}_{i=1}\, LC_{\text{group member}_i}) \; || \; Finalize$

where $||^{n}_{i=1} LC_i$ denotes the parallel composition of n copies of life-cycle LC.

The life-cycle model is used for

- software architecture
- testing
- design of user interfaces

Software lifecycle – Validation:

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

scenario	**life-cycle expression**
sdAdd	$LC_{\text{group member}}$
sdAnswer	$LC_{\text{group member}}$
sdAccess	$LC_{\text{group member}}$
sdFinalize	$LC_{\text{collaborative calendar}}$

- For the biddable domain GroupMember exactly one life-cycle exists, namely $LC_{\text{group member}}$

- The life-cycles are consistent with the state predicates in Step A3: Abstract software specification:
  - Add has no state predicates at the beginning. Hence, it can be executed an arbitrary number of times.
  - Answer can be executed if an appointment object is created beforehand. Otherwise, Answer returns an empty set and no possible appointment can be selected.
  - Access has no state predicates at the beginning and the End. Hence, it can be executed an arbitrary number of times.
  - Finalize has no state predicates at the beginning. Hence, it can be executed an arbitrary number of times.

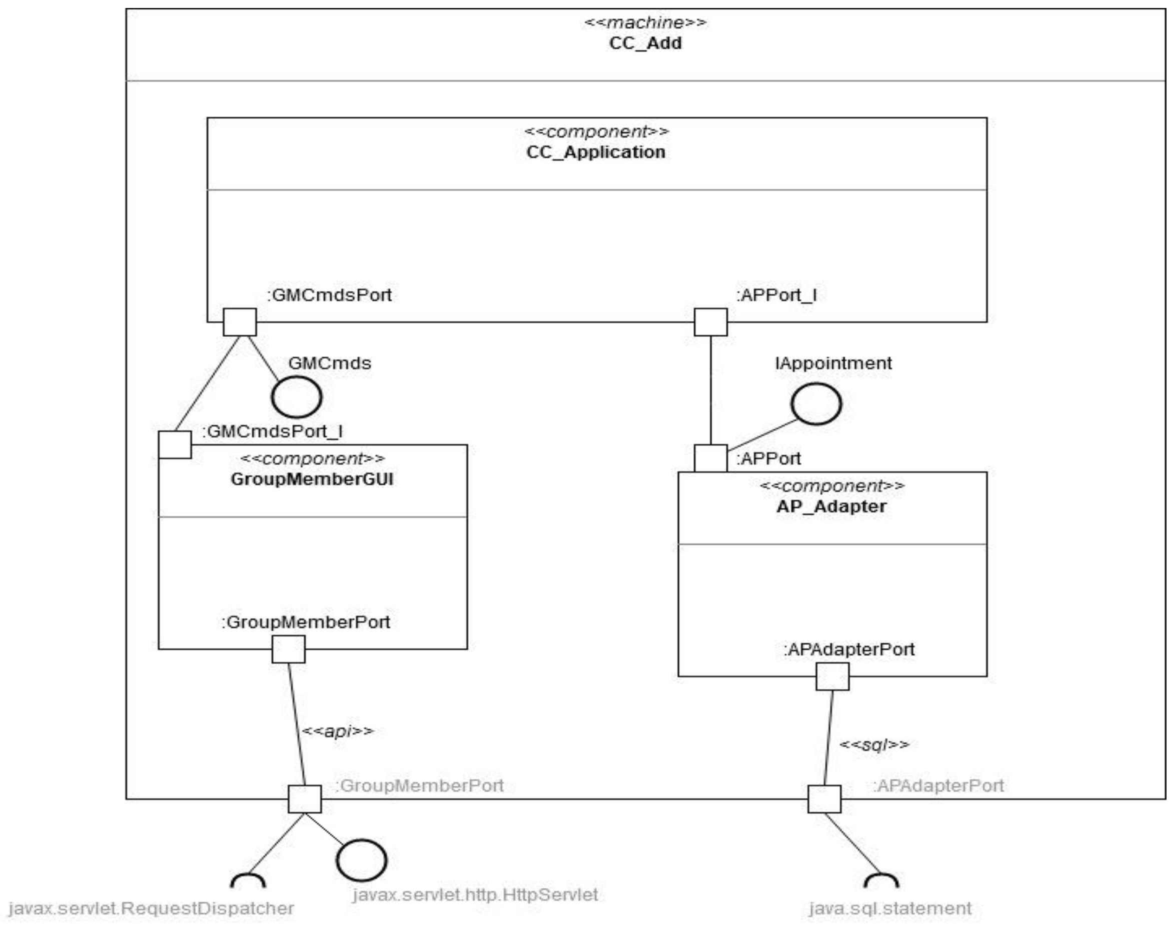
- the life-cycles are consistent with the pre- and postconditions in Step A5: Operations and data specification:
  - The sequence diagram Add contains the operation CreatNewAppointment. It has no precondition. Hence, it can be executed at any position of the life-cycle.
  - The sequence diagram Access contains the operation MakeAccessCalendar. It has no precondition. Hence, it can be executed at any position of the life-cycle.
  - The sequence diagram Answer contains the operation CreatPossibleAppontment. CreatPossibleAppontment requires, that an appintment with the supplied aid exists. This is ensured by the postcondition of CreatNewAppointment, that returns a subset of all existing appointment. Only the hid being an element of this list can be used as an input for CreatPossibleAppontment. Hence, Access must be executed before Add and Answer. And The sequence diagram Answer also contains the operation CreatSuggestedAppontment. It can be executed when an appointment with the supplied aid not exists.
  - The sequence diagram Finalize contains the operation DeadlineOfRequest. It has no precondition. Hence, it can be executed at any position of the life-cycle.

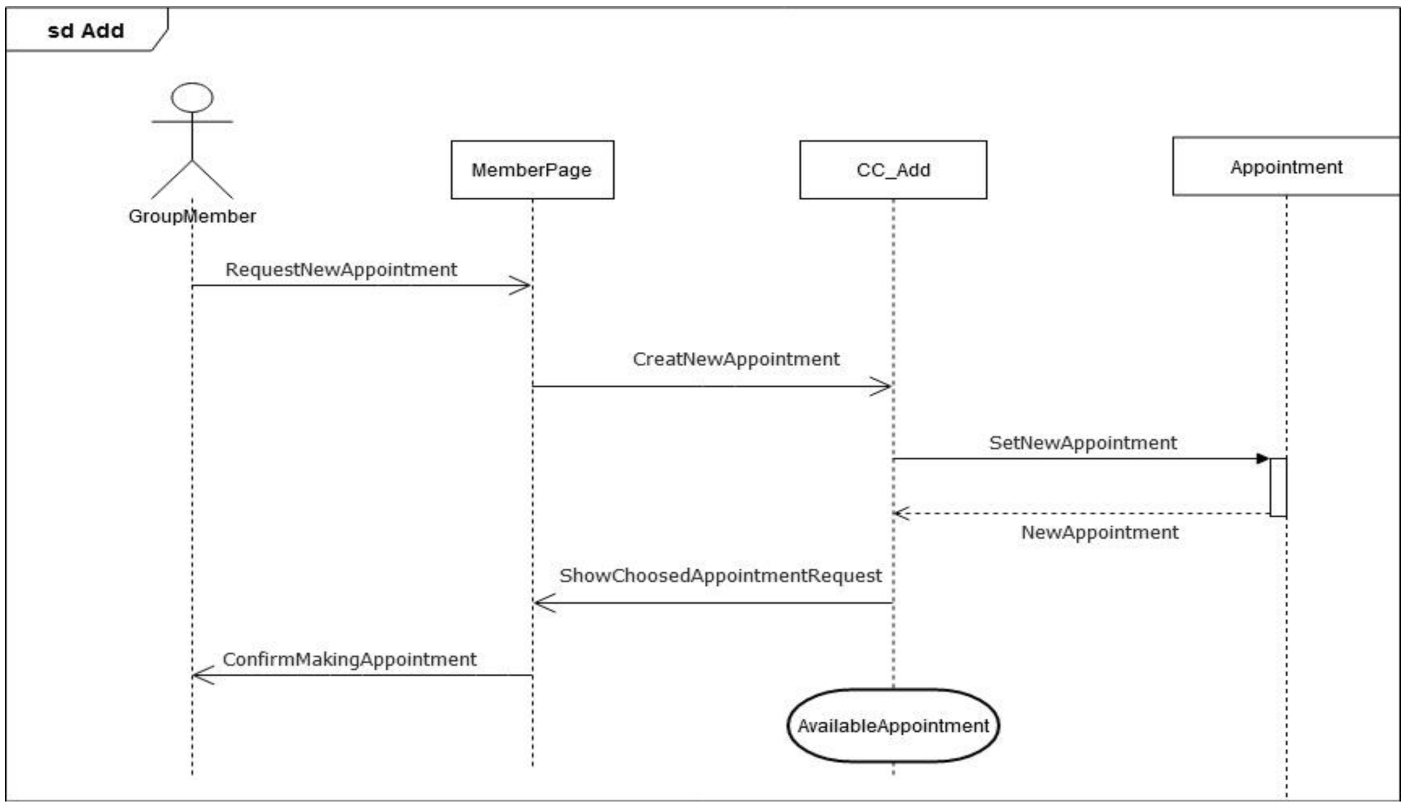
- Exactly one life-cycle exists for the machine domain, that combines all life-cycles

  The life-cycle $LC_{\text{collaborative calendar}}$ exists for the machine domain. It combines all life-cycles.

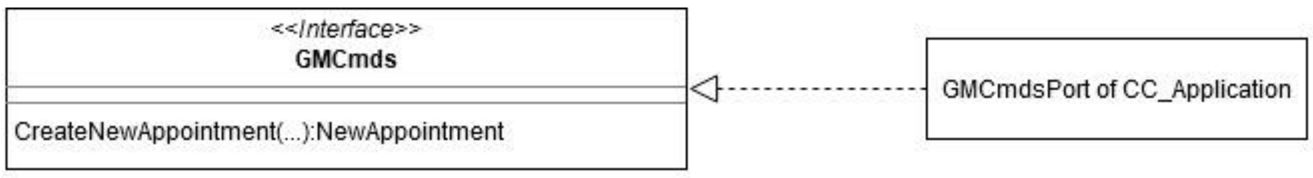
Subproblem architectures :

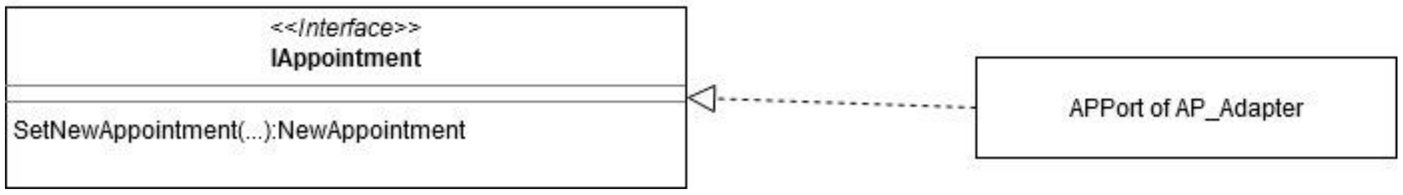
1. **CC_ADD fits to update (2).** Instantiated architectural Pattern for CC_Add :





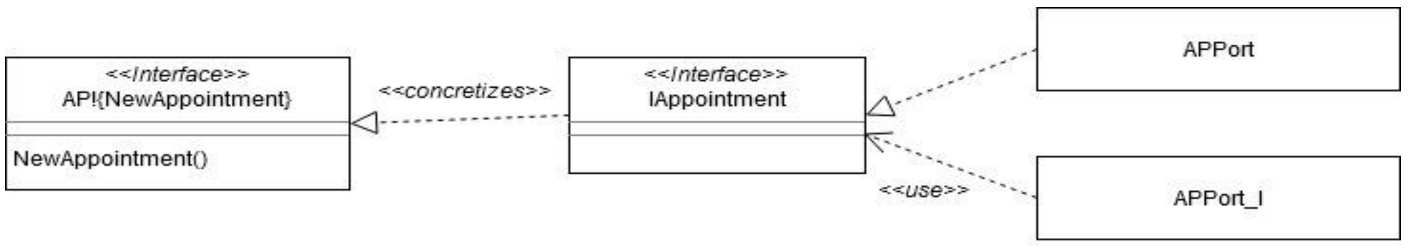
Internal interfaces in CC_Add:

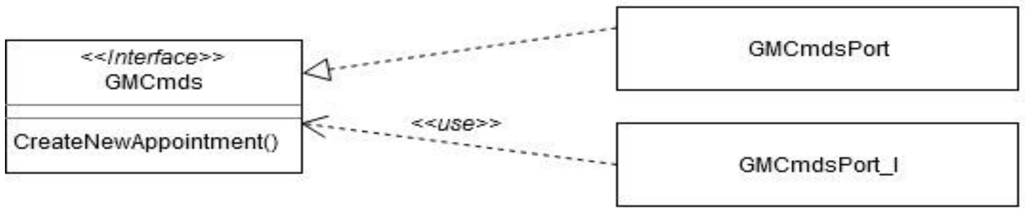

<<Interface>>
GMCmds
CreateNewAppointment(...):NewAppointment
GMCmdsPort of CC_Application

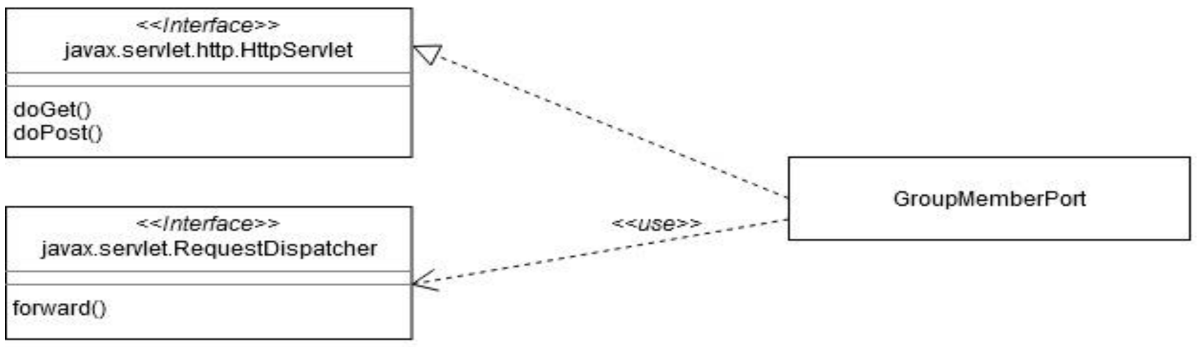


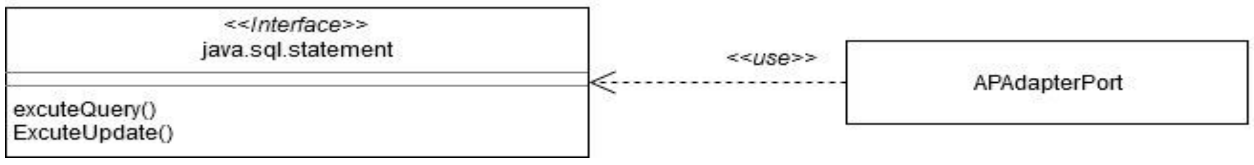
ShowChoosedAppointmentRequest realized by the return value NewAppointment of get_NewAppointment.

Port types and interfaces relations for CC_Add :

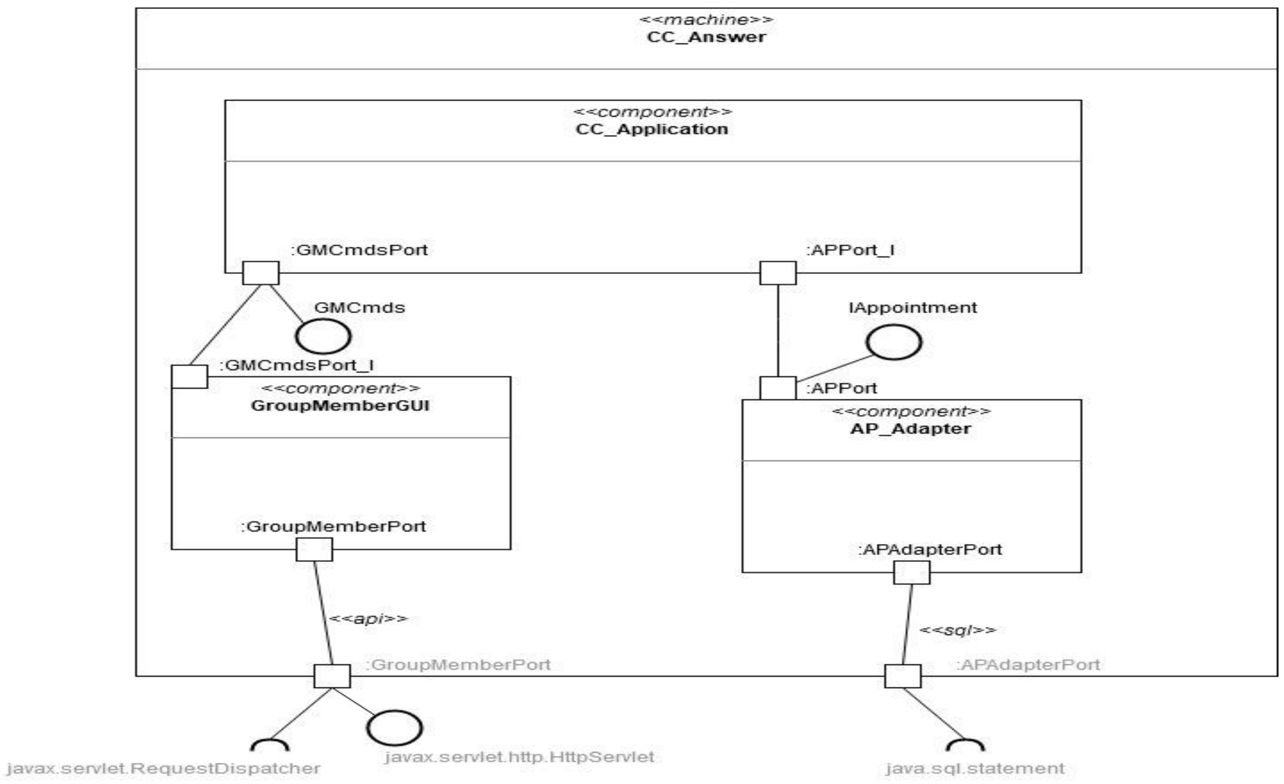


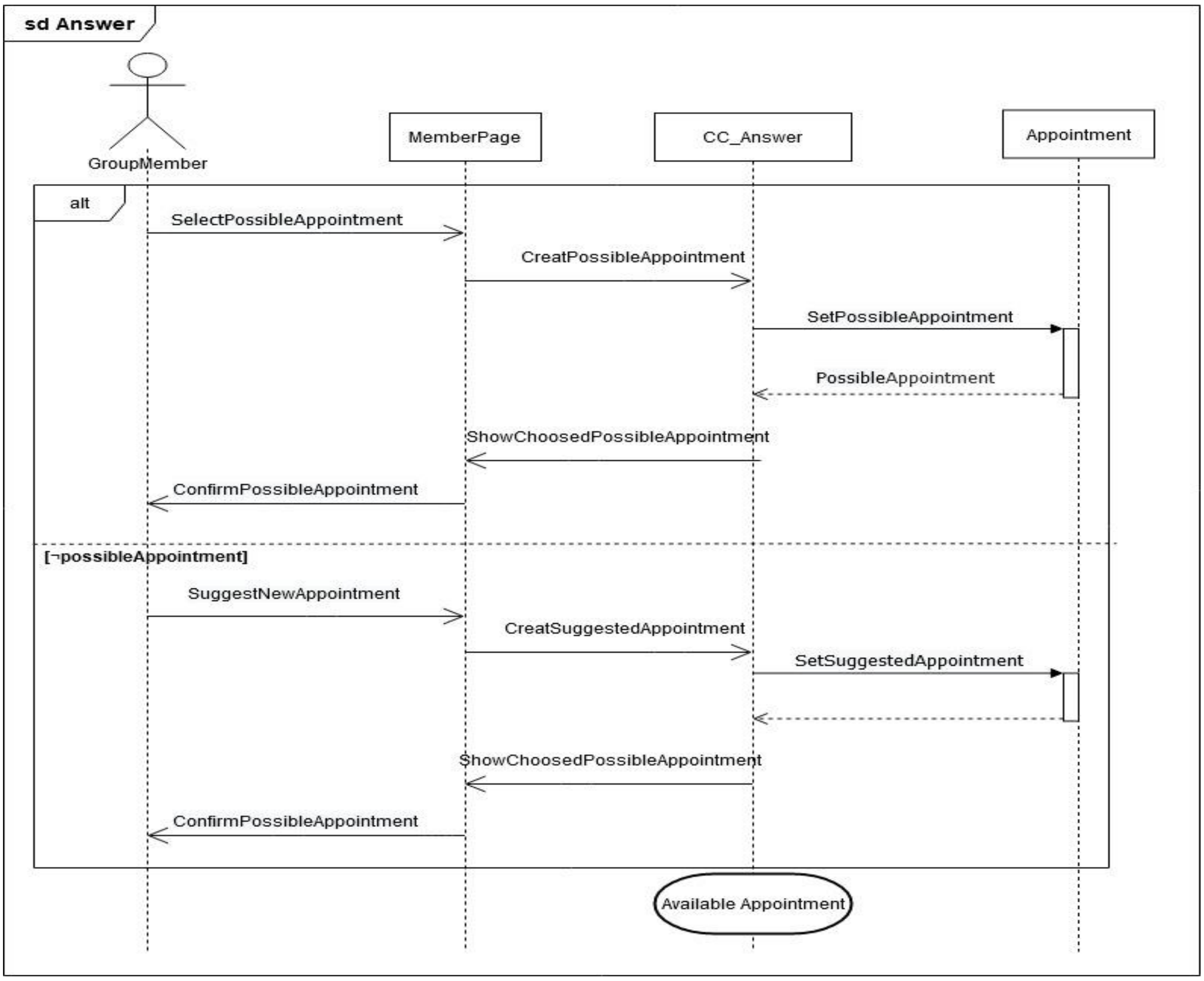




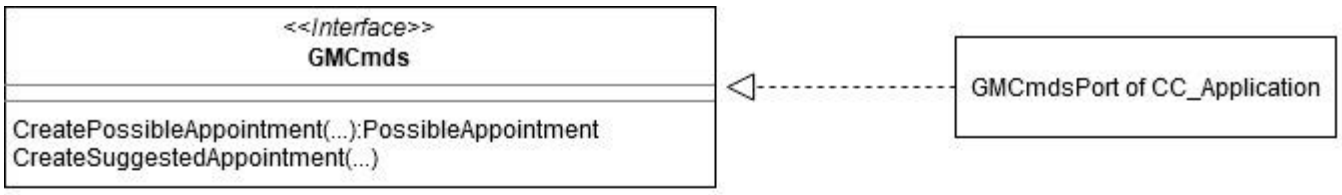


2. CC_Answer fits to update (2). Instantiated architectural Pattern for CC_Answer :

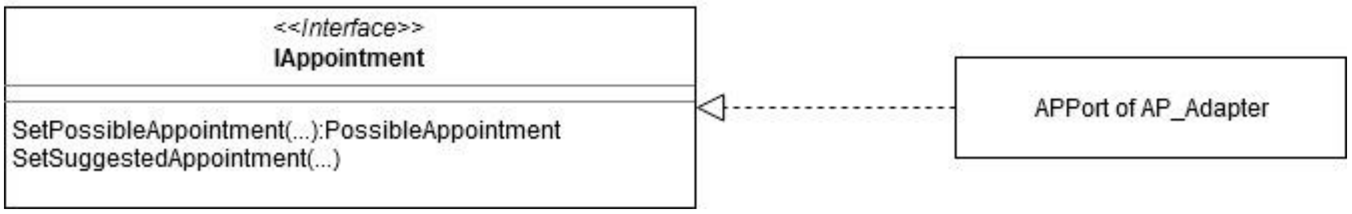




Internal interfaces in CC_Answer:



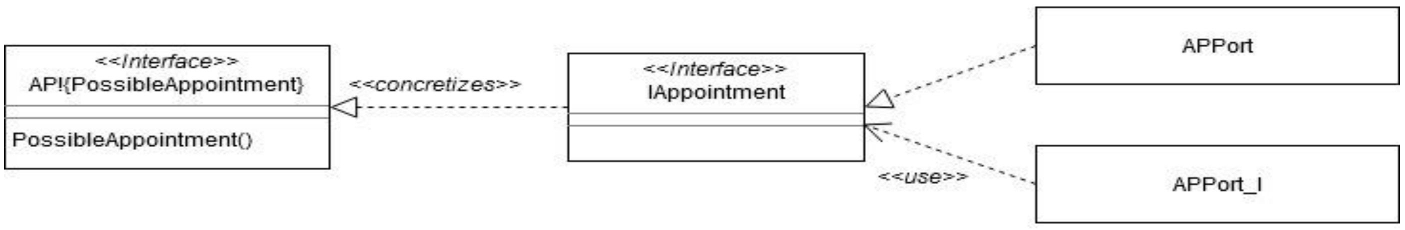
<<Interface>>
GMCmds
CreatePossibleAppointment(...):PossibleAppointment
CreateSuggestedAppointment(...)
GMCmdsPort of CC_Application

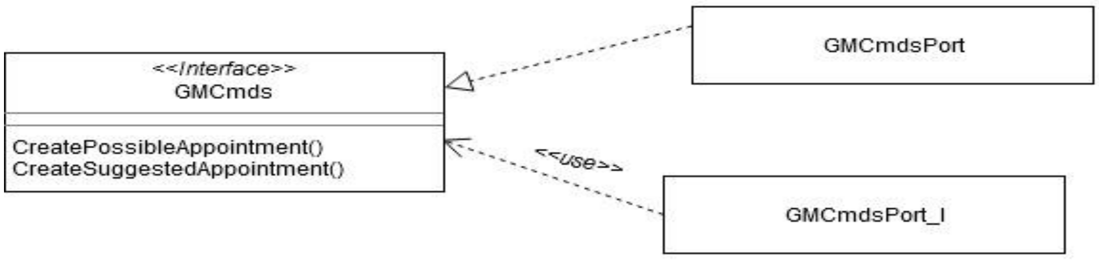


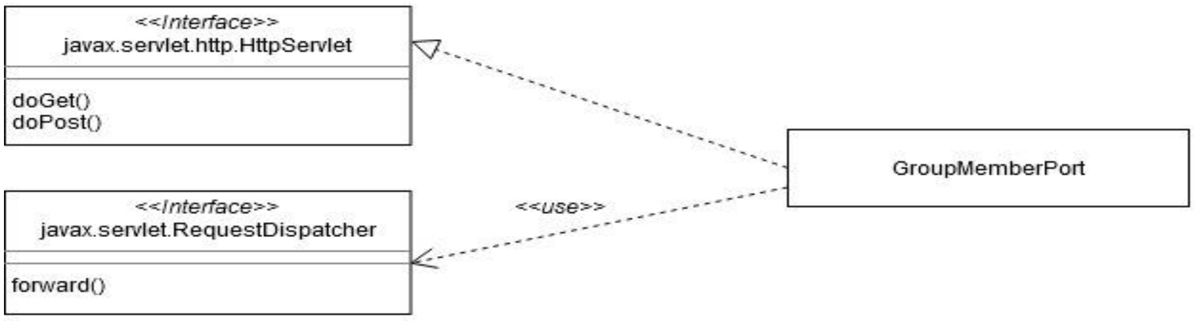
<<Interface>>
IAppointment
SetPossibleAppointment(...):PossibleAppointment
SetSuggestedAppointment(...)
APPort of AP_Adapter

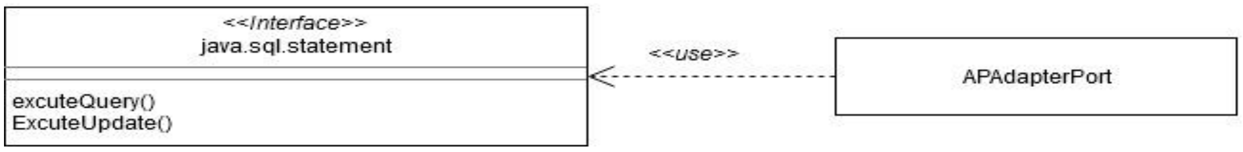
ShowChoosedPossibleAppointment realized by the return value PossibleAppointment of get_PossibleAppointment.

Port types and interfaces relations for CC_Answer :

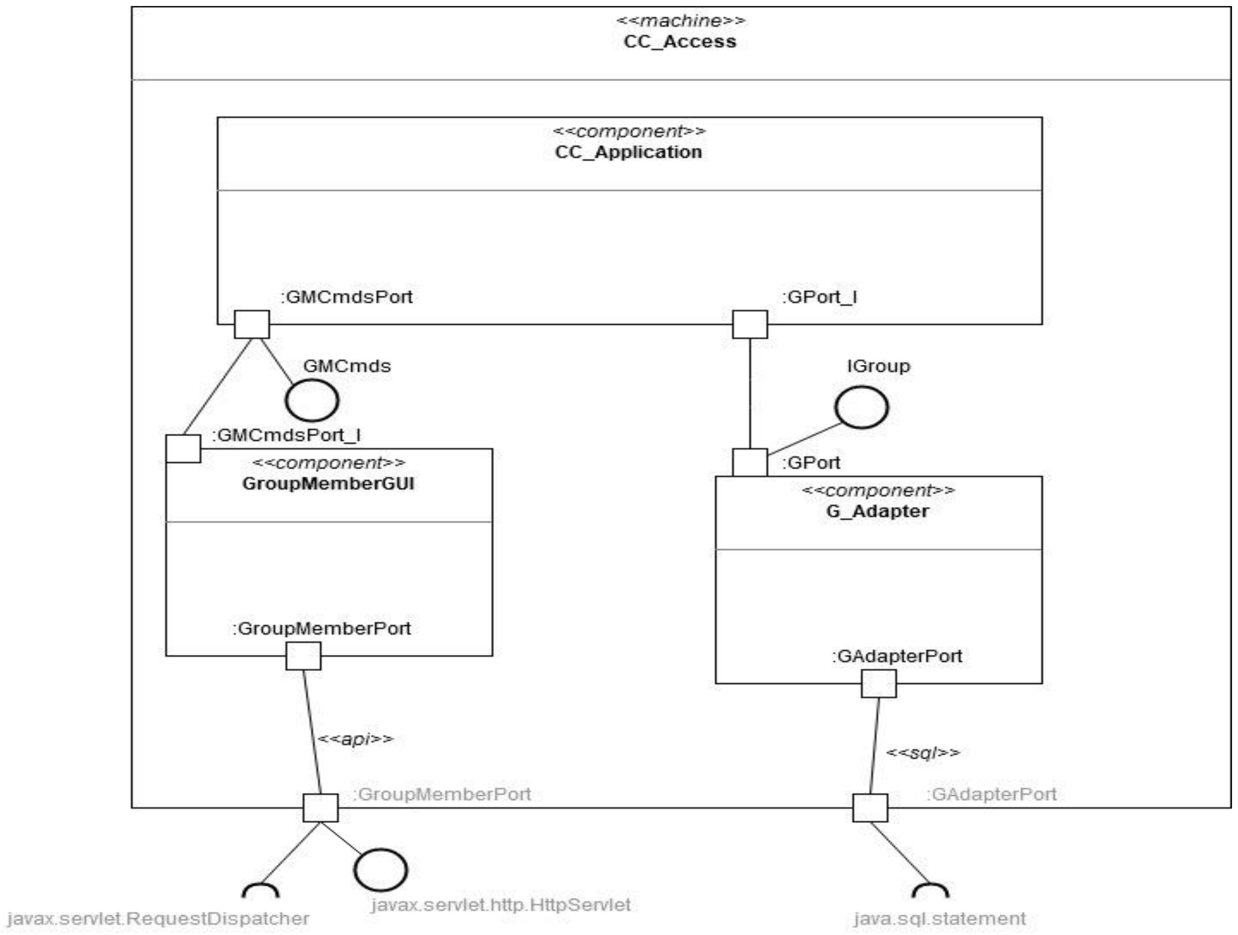


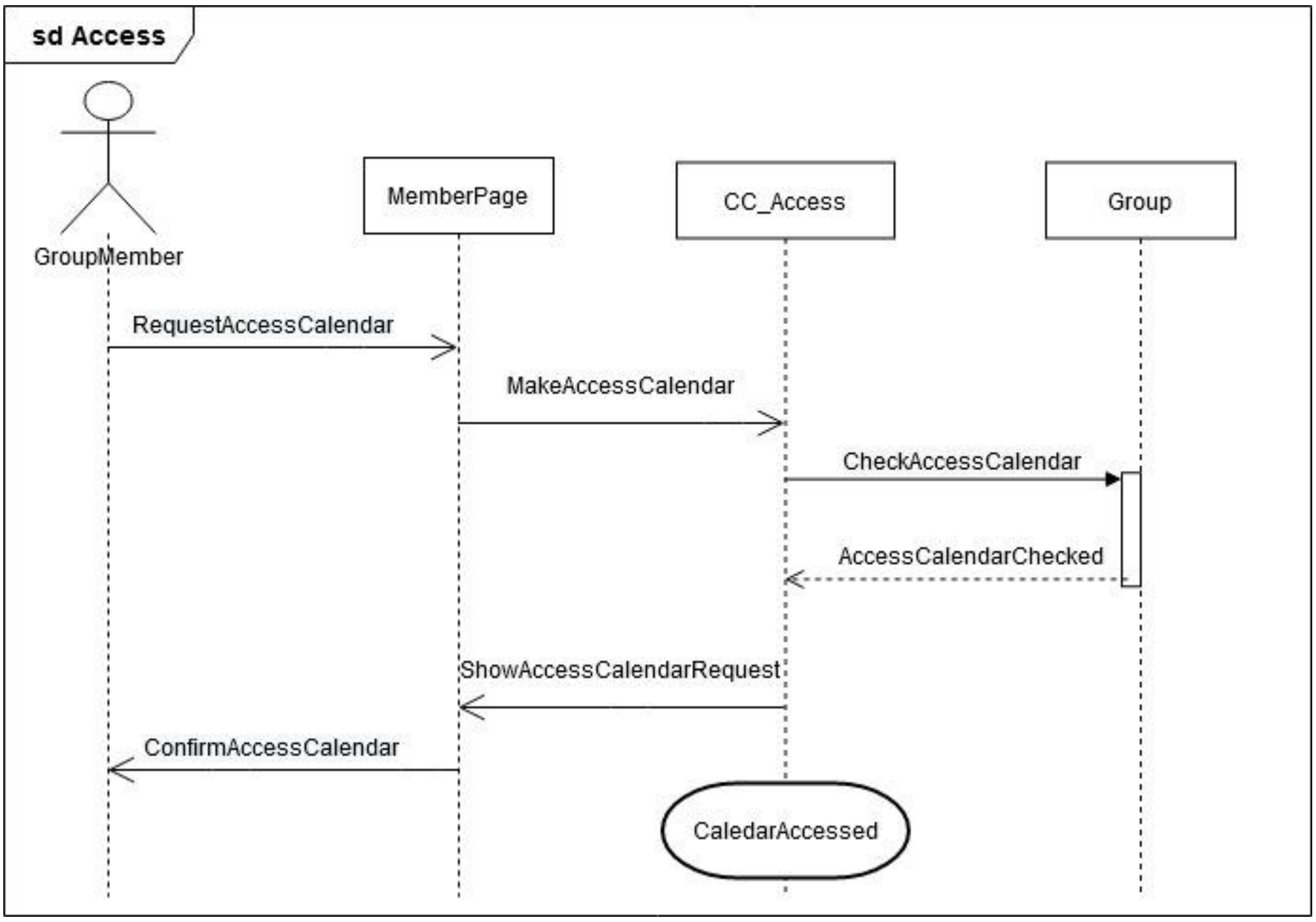




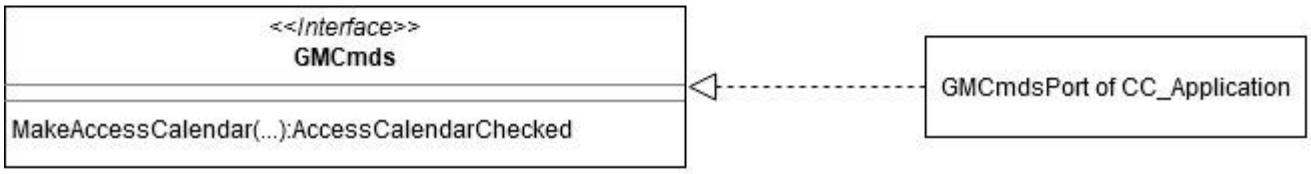


3. CC_Access fits to query (2). Instantiated architectural Pattern for CC_Access :

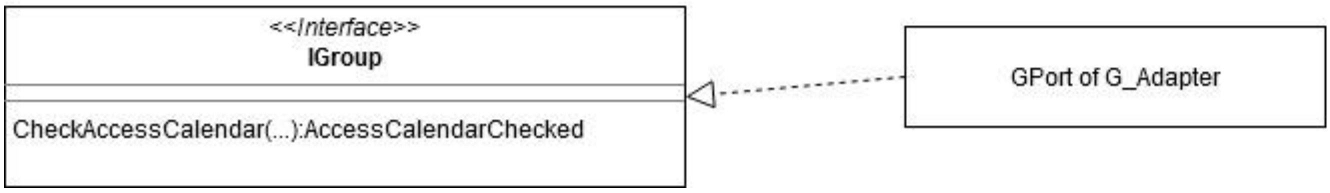




Internal interfaces in CC_Access:

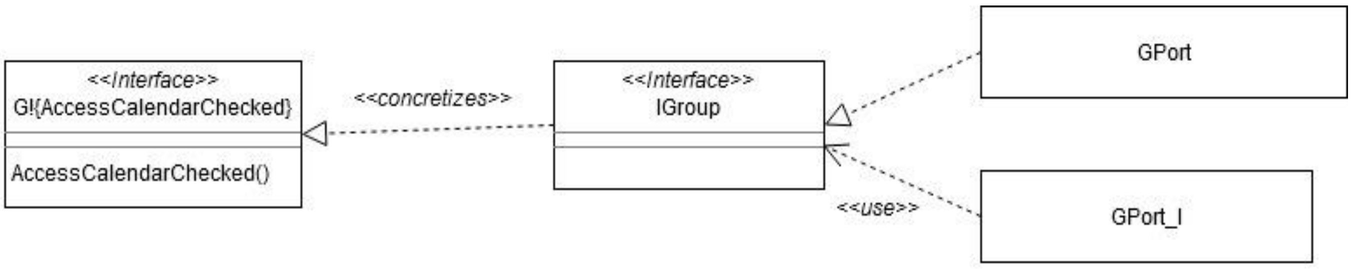


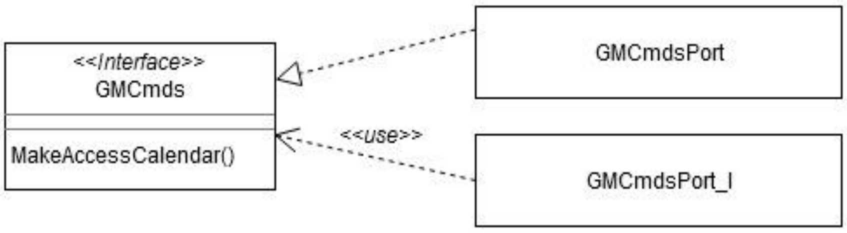
<<Interface>>
GMCmds
MakeAccessCalendar(...):AccessCalendarChecked
GMCmdsPort of CC_Application

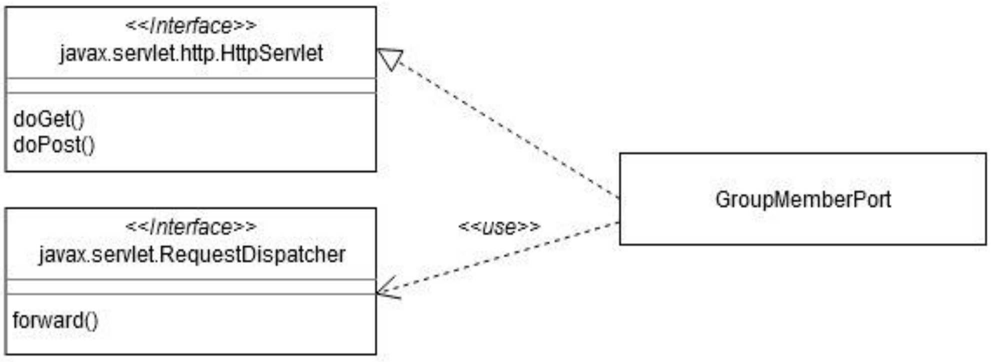


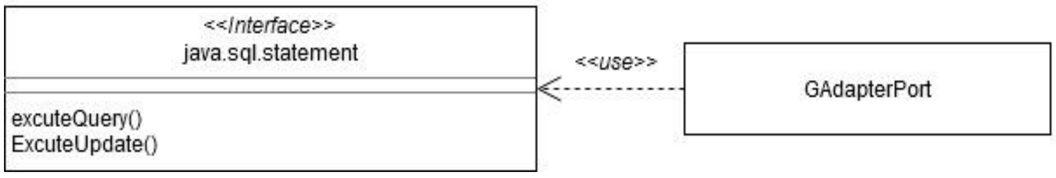
ShowAccessCalendarRequest realized by the return value AccesCalendarChecked of CheckAccessCalendar.

Port types and interfaces relations for CC_Access :

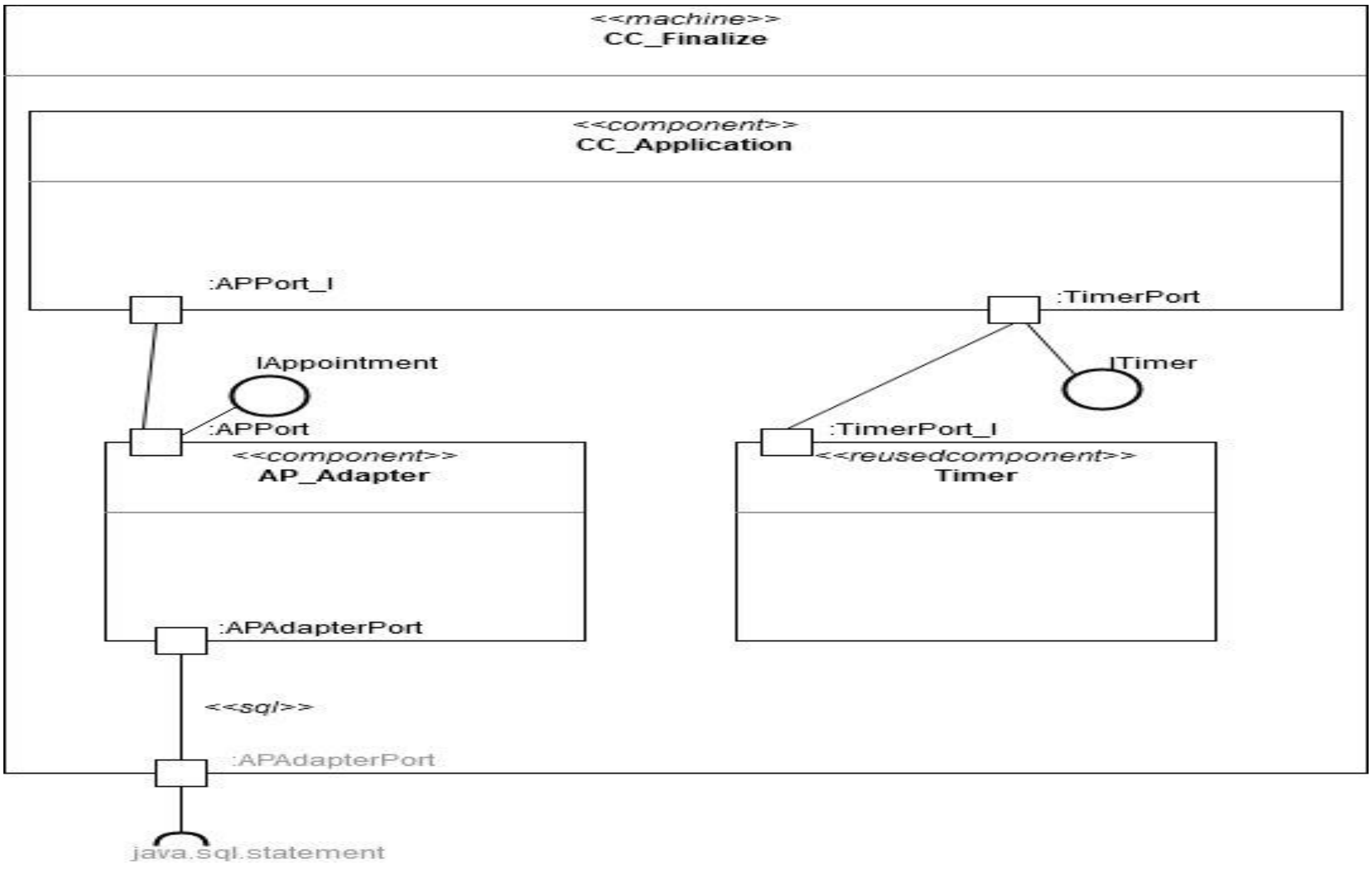


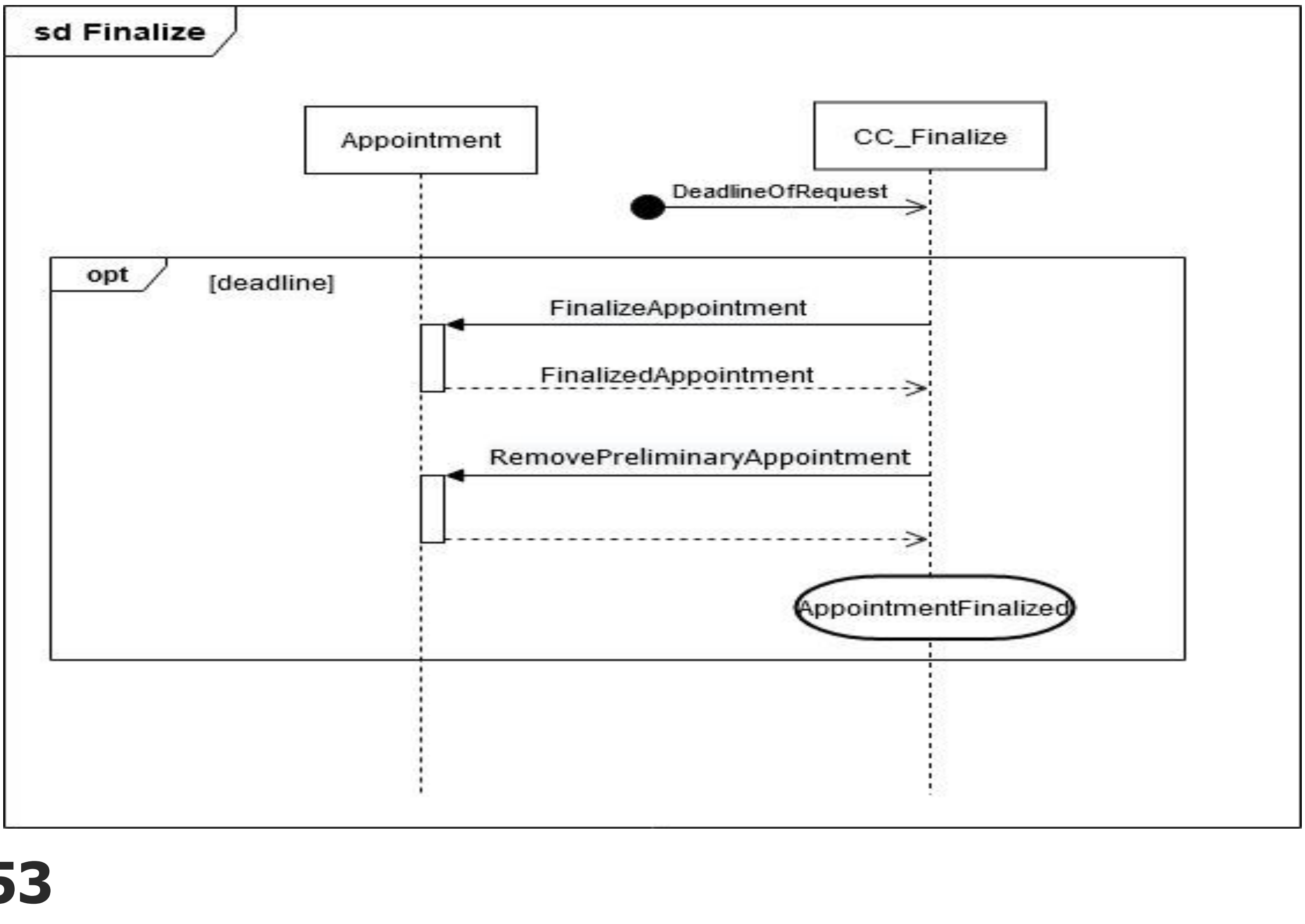






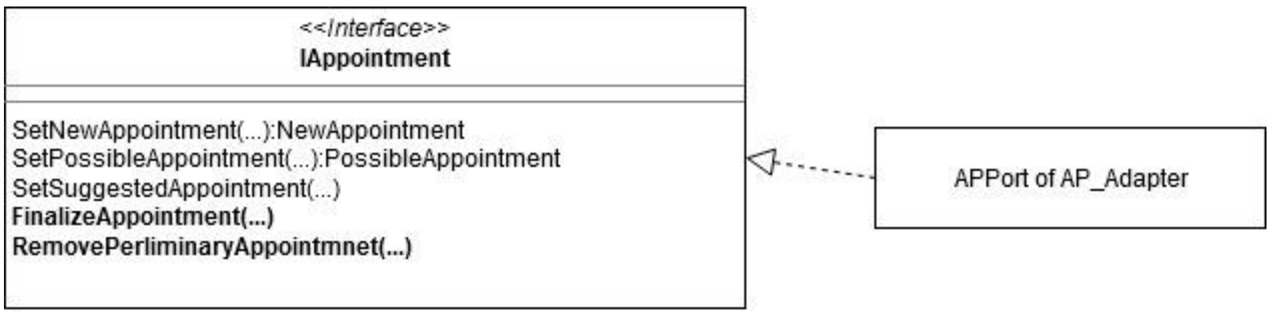
4. **CC_Finalize fits to transformation + simple transformation.** Instantiated architectural Pattern for CC_Finalize :

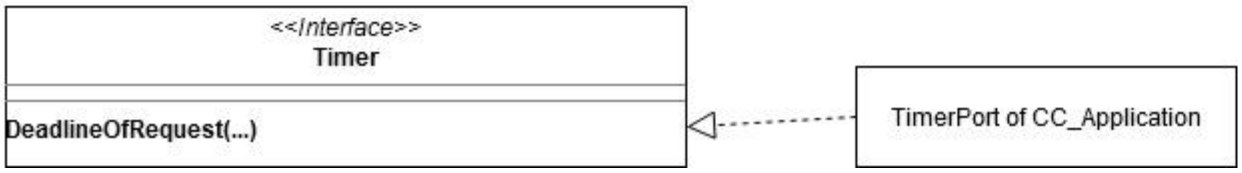




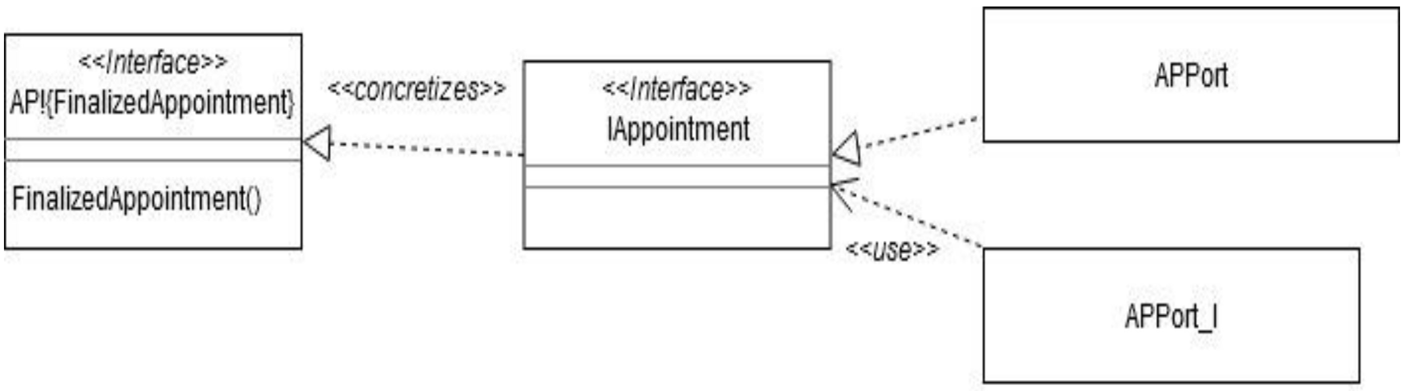
Internal interfaces in CC_Finalize:

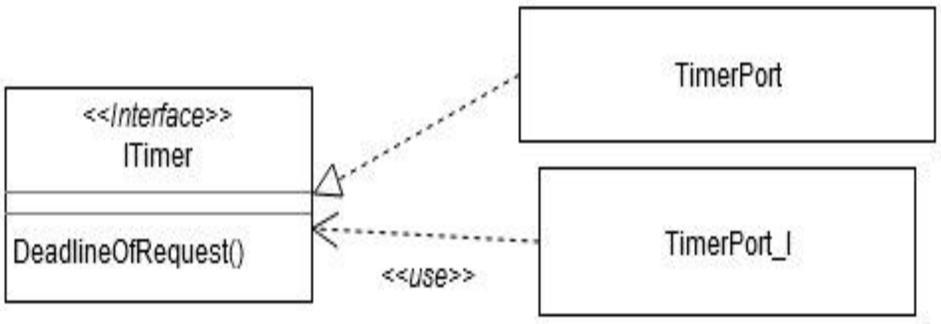
IGroup from CC_Access and IAppointment from both CC_Add and CC_Answer are extended as follows:

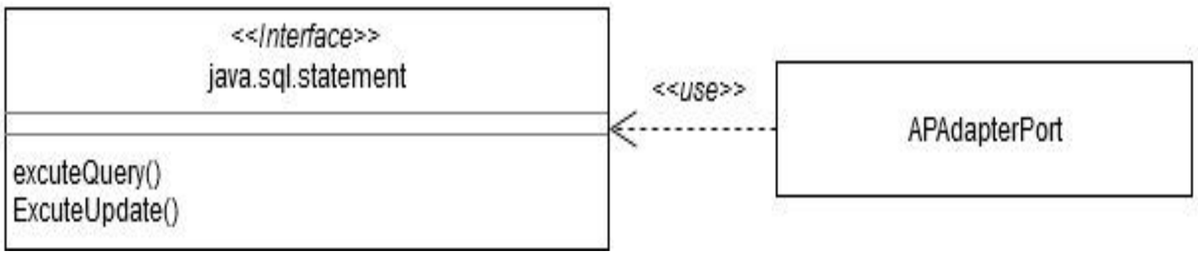

<<Interface>>
IAppointment
SetNewAppointment(...):NewAppointment
SetPossibleAppointment(...):PossibleAppointment
SetSuggestedAppointment(...)
FinalizeAppointment(...)
RemovePerliminaryAppointmnet(...)
APPort of AP_Adapter



Port types and interfaces relations for CC_Finalize :

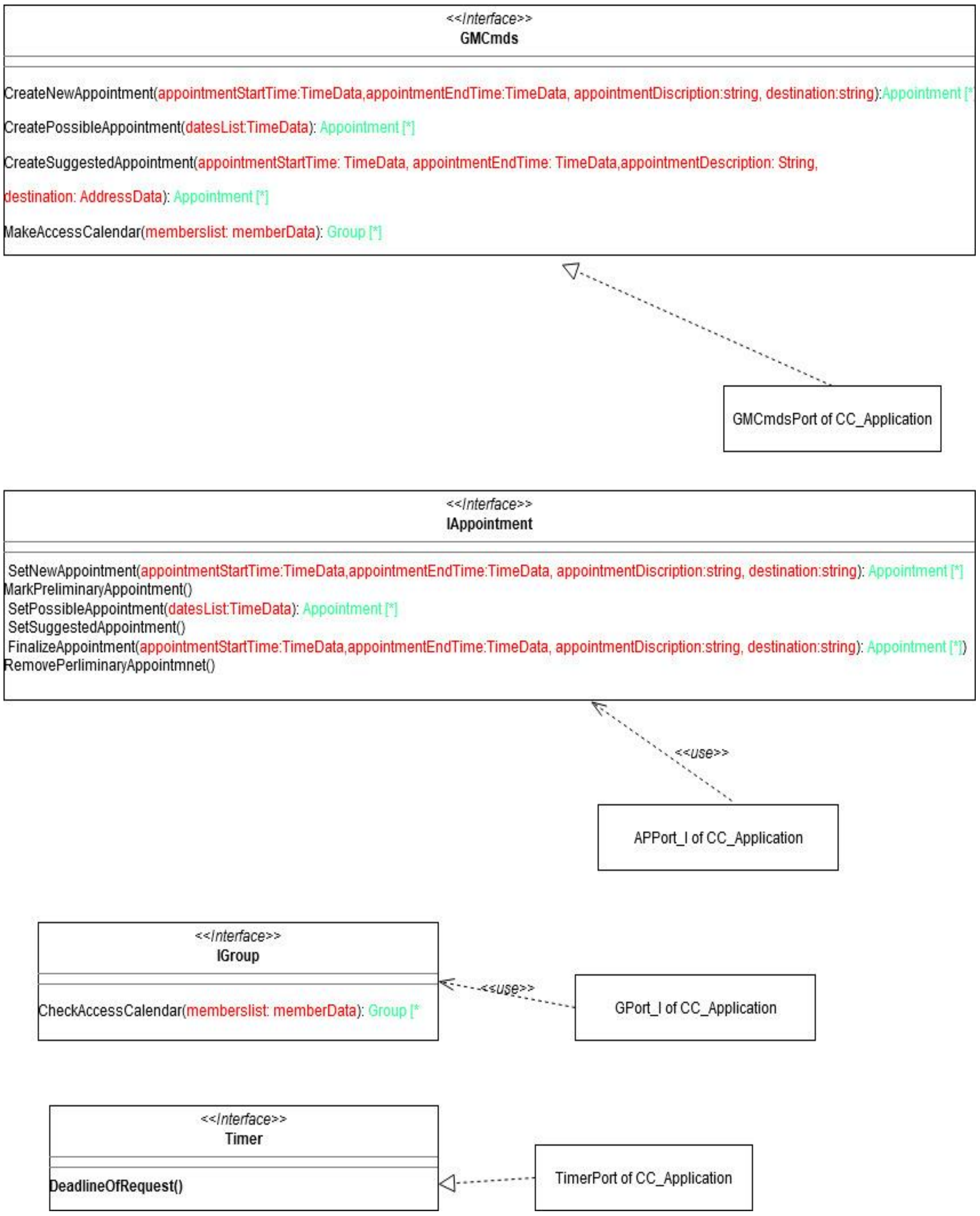






Refining app_if interface Classes :

By abstract specifications and operation specification


<<Interface>>
GMCmds
CreateNewAppointment(appointmentStartTime:TimeData,appointmentEndTime:TimeData, appointmentDiscription:string, destination:string):Appointment [*
CreatePossibleAppointment(datesList:TimeData): Appointment [*]
CreateSuggestedAppointment(appointmentStartTime: TimeData, appointmentEndTime: TimeData,appointmentDescription: String,
destination: AddressData): Appointment [*]
MakeAccessCalendar(memberslist: memberData): Group [*]
GMCmdsPort of CC_Application
<<Interface>>
IAppointment
SetNewAppointment(appointmentStartTime:TimeData,appointmentEndTime:TimeData, appointmentDiscription:string, destination:string): Appointment [*]
MarkPreliminaryAppointment()
SetPossibleAppointment(datesList:TimeData): Appointment [*]
SetSuggestedAppointment()
FinalizeAppointment(appointmentStartTime:TimeData,appointmentEndTime:TimeData, appointmentDiscription:string, destination:string): Appointment [*])
RemovePerliminaryAppointmnet()
<<use>>
APPort_I of CC_Application
<<Interface>>
IGroup
CheckAccessCalendar(memberslist: memberData): Group [*
<<use>>
GPort_I of CC_Application
<<Interface>>
Timer
DeadlineOfRequest()
TimerPort of CC_Application

Refining tech_if" interface classes:

Considered interface in subproblem architecture	**technical interface**
<<api>> javax.servlet.http.HttpServlet in CC Add	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.http.HttpServlet in CC Answer	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.http.HttpServlet in CC Access	<<api>> AT!{doGet, doPost}
<<api>> javax.servlet.RequestDispatcher in CC Add	<<api>> CC!{forward}
<<api>> javax.servlet.RequestDispatcher in CC Answer	<<api>> CC!{forward}
<<api>> javax.servlet.RequestDispatcher in CC Access	<<api>> CC!{forward}
<<sql>> java.sql.Statement in CC Add	<<call return, sql>>CC!{executeQuery, executeUpdate}
<<sql>> java.sql.Statement in CC Answer	<<call return, sql>>CC!{executeQuery, executeUpdate}
<<sql>> java.sql.Statement in CC Access	<<call return, sql>>CC!{executeQuery, executeUpdate}
<<sql>> java.sql.Statement in CC finalize	<<call return, sql>>CC!{executeQuery, executeUpdate}

Refining adapter_if' interface classes:

There are no HAL components is the subproblem architectures. Hence, there are no adapter if' interface classes that need to be refined.

Merged architecture:

The components can be merged as follows:

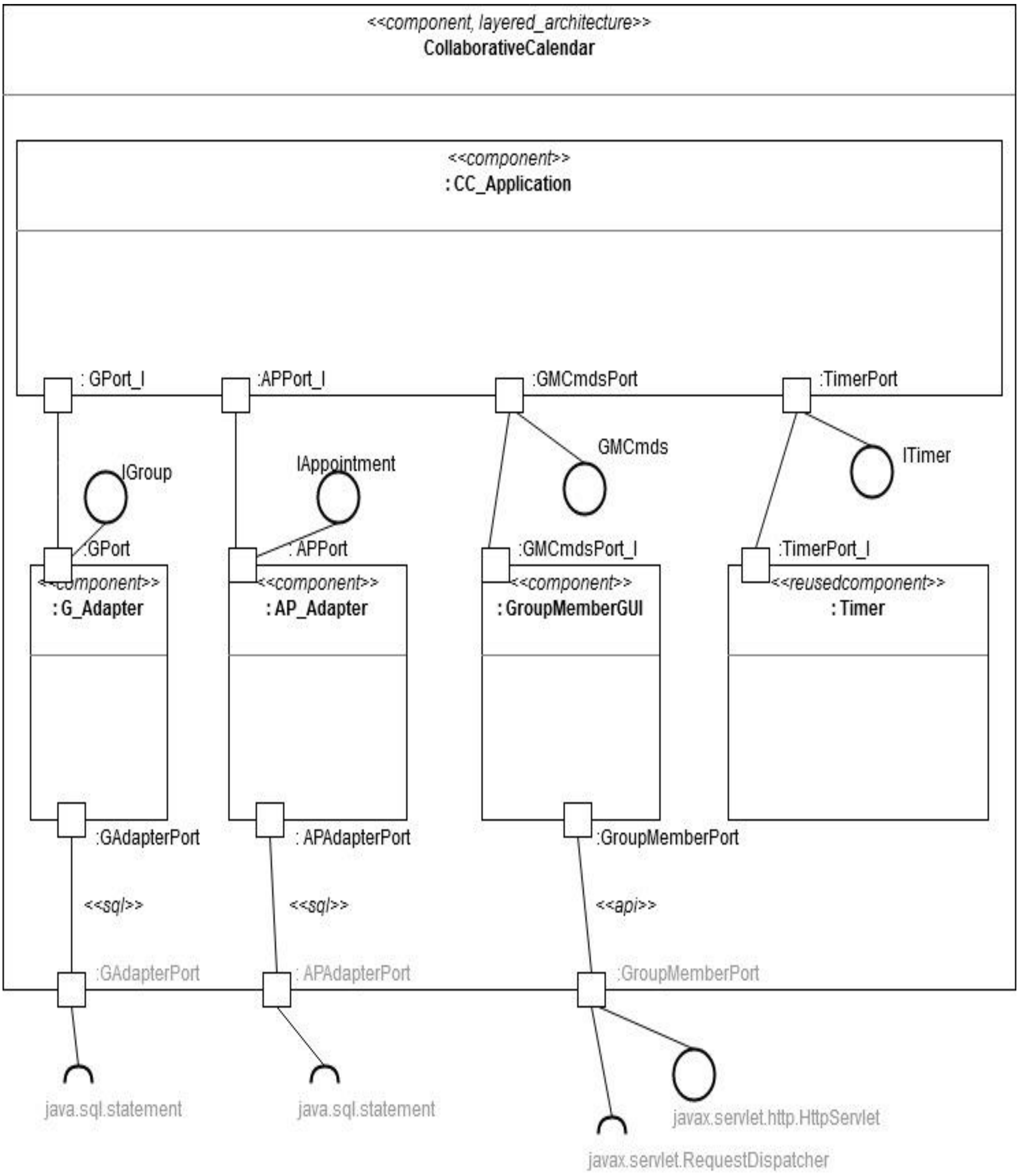
- The application components in all architectures for the subproblems should be merged because the Subproblem Add, Answer and Access are related alternatively. The Subproblem Finalize is also merged because of reasons of simplicity.
- The AP Adapters in all architectures for the subproblems should be merged because it is an adapter, establishing the connection to the DB.
- The G 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 in only 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 Finalize sequence.

Components to Develop

- We have to develop the CC Application, a GroupMemberGUI, and adapters for the external components.
- The AP Adapter is responsible to create and maintain. tables for all persistent classes.
- The G Adapter is responsible to create and maintain. tables for all persistent classes.



D1: Software architecture Validation:

- All messeges of Step A3: Abstract software specification are interfaces of the application layer:

Sequence Diagram	Messege	in/out	Application Lazer Interface	Required / Provided
Add	CreateNewAppointment	In	GMCmds:: CreateNewAppointment	Provided
	get_NewAppointment	Out	IAppointment:: get_NewAppointment	Required
	NewAppointment	In	Return value of IAppointment:: get_NewAppointment	Required
	setNewAppointment	Out	IAppointment:: setNewAppointment	Required
	MarkPreliminarlyAppointment	In	IAppointment:: MarkPreliminarlyAppointment	Required
	ShowChoosedAppointmentRequest	Out	Return value of GMCmds:: CreateNewAppointment	Provided
Answer	CreatePossibleAppointment	In	GMCmds:: CreatePossibleAppointment	Provided
	CreateSuggestedAppointment	In	GMCmds:: CreatePossibleAppointment	Provided
	get_PossibleAppointment	Out	IAppointment:: get_PossibleAppointment	Required
	PossibleAppointment	In	Return value of IAppointment:: get_PossibleAppointment	Required
	ShowChoosedPossibleAppointment	Out	Return value of GMCmds:: CreatePossibleAppointment	Provided
	ShowChoosedPossibleAppointment	Out	Return value of GMCmds:: CreateSuggestedAppointment	Provided
	SetPossibleAppointment	Out	IAppointment:: SetPossibleAppointment	Required
	SetSuggestedAppointment	Out	IAppointment:: SetSuggestedAppointment	Required
Access	MakeAccessCalendar	In	GMCmds:: MakeAccessCalendar	Provided
	checkAccessCalendar	Out	IGroup:: get_AccessCalendar	Required
	AccessCalendarChecked	In	Return value of IGroup:: checkAccessCalendar	Required
	ShowAccessCalendarRequest	Out	Return value of GMCmds:: MakeAccessCalendar	Provided
Finalize	DeadlineOfRequest	In	ITimer: DeadlineOfRequest	Provided
	FinalizeAppointment	Out	IAppointment::FinalizeAppointment	Required
	RemovePreliminaryAppointment	Out	IAppointment:: RemovePreliminaryAppointment	Required

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

Provided by machine	**Required by adapter / provided by app**
javax.servlet.http.HttpServlet	GMCmds
-	ITimer

Required by machine	**Provided by adapter / required by app**
javax.servlet.RequestDispatcher	Return values in GMCmds
Java.sql.statement	IAppointment
Java.sql.statement	IGroup

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

external port type	**Interface in architecture**	**Required/ Provided**	**Interface in technical context diagram**
GroupMemberPort	javax.servlet.http.HttpServlet	provided	AT!{doGet, doPost}
	javax.servlet.RequestDispatcher	required	CC! {forward}
APAdapterPort	java.sql.Statement	required	CC!{executeQuery, executeUpdate}
GAdapterPort	java.sql.Statement	required	CC!{executeQuery, executeUpdate}

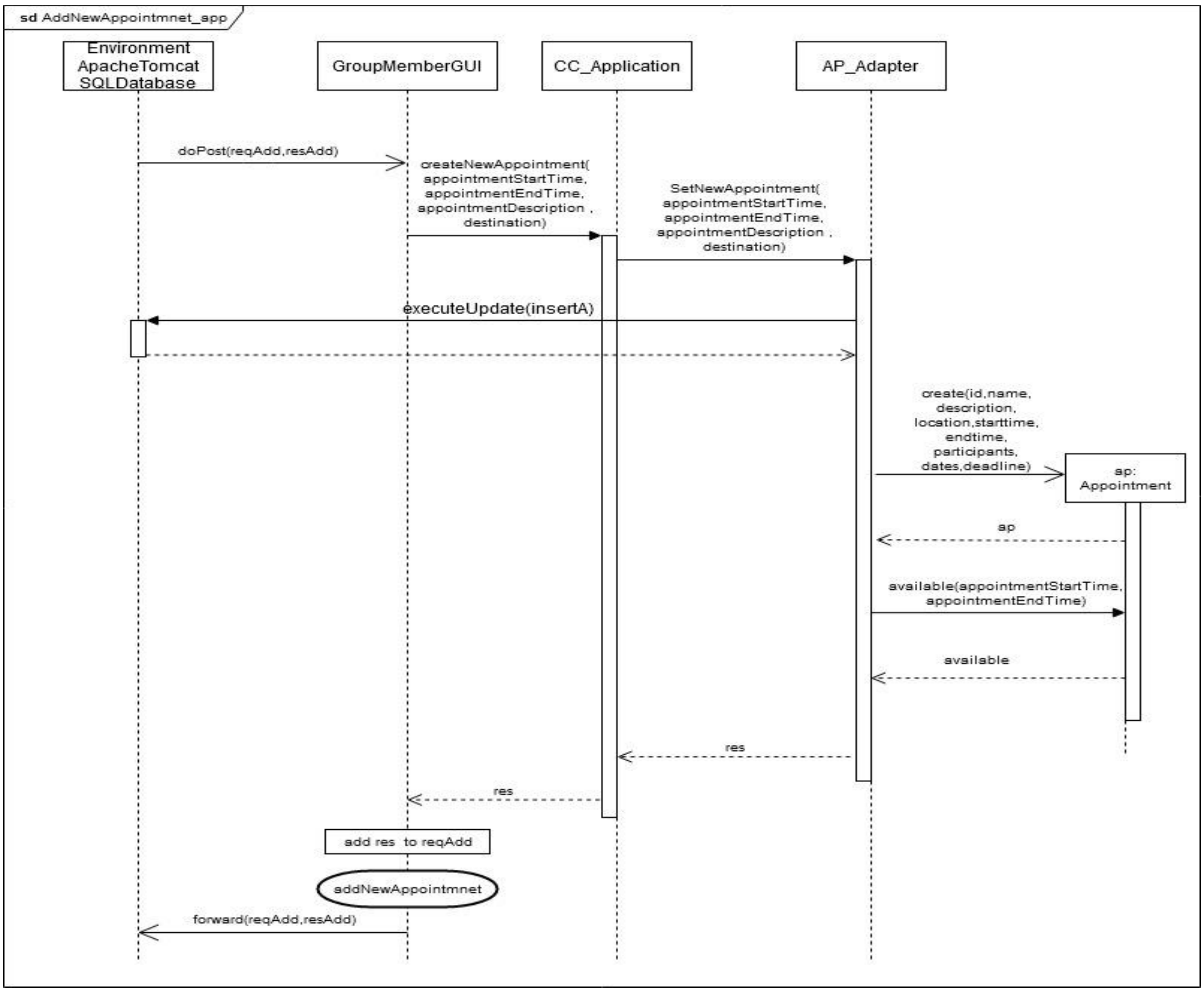
Inter-component interaction

CreateNewAppointment – Operation specification:

OCL constraint:

```
context CC_Add :: CreatNewAppointment( appointmentStartTime : TimeData,
    appointmentEndTime : TimeData, appointmentDescription : String, destination:AddressData)
Pre: true
Post: let res: Set (Appointment) = ap ->one (a: Appointment |
            a.startTime = appointmentStartTime and
            a.endTime = appoimtentEndTime  and
            a.available (appointmentStartTime, appoimtentEndTime )) ->asSet()
      in
            MemberPage^ShowChoosedAppointmentRequest( res )
```

sdAddNewAppointmnet_app:



sd AddNewAppointmnet_app
Environment ApacheTomcat SQLDatabase
GroupMemberGUI
CC_Application
AP_Adapter
doPost(reqAdd,resAdd)
createNewAppointment(appointmentStartTime, appointmentEndTime, appointmentDescription , destination)
SetNewAppointment(appointmentStartTime, appointmentEndTime, appointmentDescription , destination)
executeUpdate(insertA)
create(id,name, description, location,starttime, endtime, participants, dates,deadline)
ap: Appointment
ap
available(appointmentStartTime, appointmentEndTime)
available
res
res
add res to reqAdd
addNewAppointmnet
forward(reqAdd,resAdd)

CreateNewAppointment – Remarks:

- reqAdd represents a HTTPServletRequest object containing the required user input.
- resAdd represents a HttpServletResponse object as the counterpart for the request.
- The state predicate AvailableApointment represents that the list of available Appointment is shown.
- ap refers to an object of class Appointment.
- available(…) checks whether the appointment is already available.
- res is a set of available appointments that fit to the selection criteria. The set will be added to the request object to be processed by the server.
- 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.

```
insertA :
INSERT INTO  Appointment
  ( id,name, description,location,startTime,endTime)  values

("aid","name","description","location","startTime","endTime")
```

Inter-component interaction

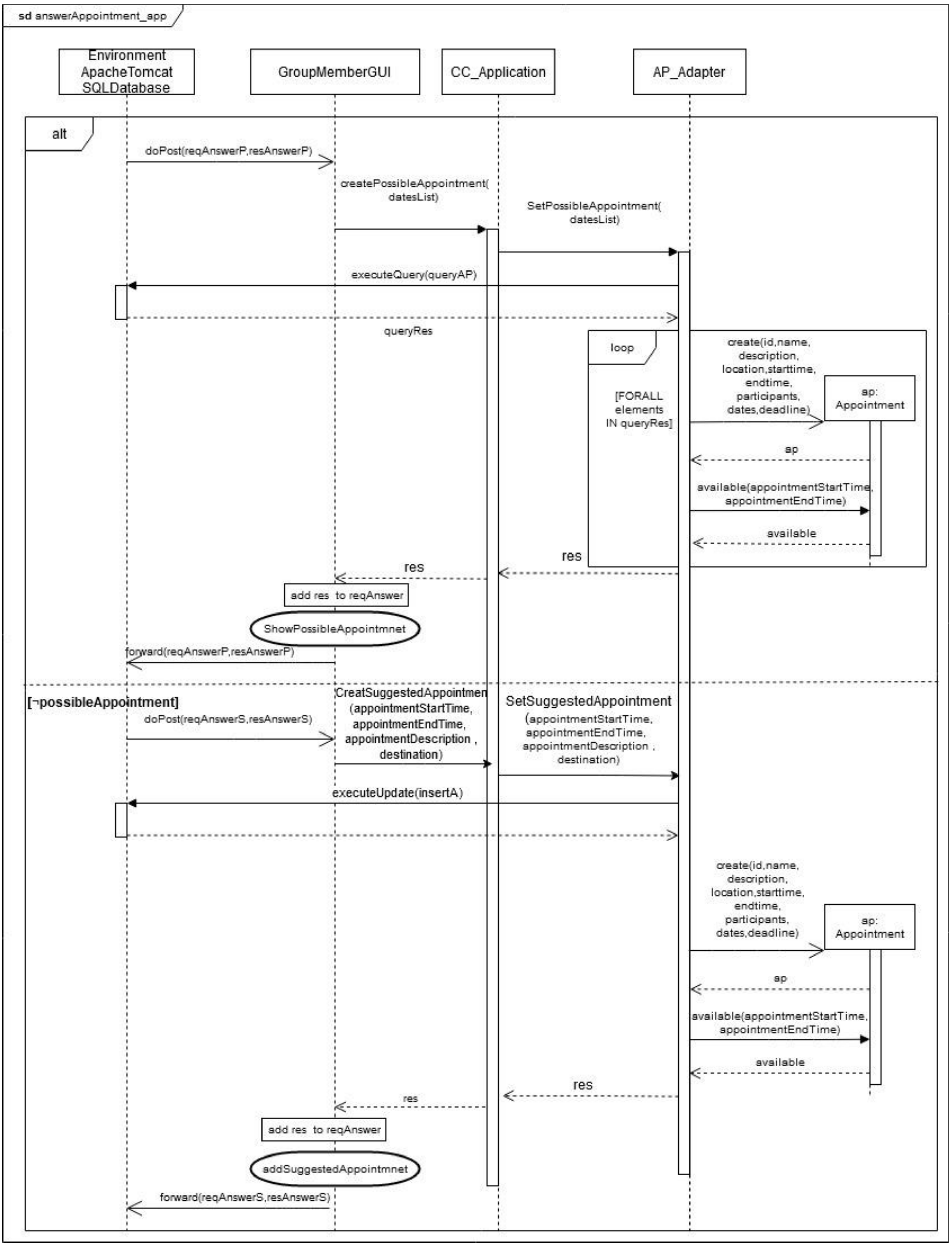
CreatSuggestedAppointment / CreatPossibleAppointment – Operation specification :

OCL constraint:

```
context CC_Answer :: CreatPossibleAppointment(appointmentStartTime : TimeData,
    appointmentEndTime : TimeData, appointmentDescription : String, destination :
    AddressData).
Pre: true
Post: let res: Set (Appointment) = ap -> select (a: Appointment |
            a.id = aid and
             a.available (appointmentStartTime, appoimtentEndTime )) ->asSet() and
        in
             MemberPage ^ ShowChoosedPossibleAppointment(res)
```

```
Context CC_Answer :: CreatSuggestedAppointment(datesList: TimeData)
Pre: true
Post: let res: Set (Appointment) = ap ->one (b: Appointment |
             b.startTime = appointmentStartTime and
             b.endTime = appoimtentEndTime and
             b.available (appointmentStartTime, appoimtentEndTime )) ->asSet() and
        in
             MemberPage ^ ShowChoosedPossibleAppointment(res)
```

sdAnswerAppointment_app:


sd answerAppointment_app
Environment ApacheTomcat SQLDatabase
GroupMemberGUI
CC_Application
AP_Adapter
alt
doPost(reqAnswerP,resAnswerP)
createPossibleAppointment(datesList)
SetPossibleAppointment(datesList)
executeQuery(queryAP)
queryRes
loop
[FORALL elements IN queryRes]
create(id,name, description, location,starttime, endtime, participants, dates,deadline)
ap: Appointment
ap
available(appointmentStartTime, appointmentEndTime)
available
res
res
add res to reqAnswer
ShowPossibleAppointmnet
forward(reqAnswerP,resAnswerP)
[¬possibleAppointment]
doPost(reqAnswerS,resAnswerS)
CreatSuggestedAppointmen (appointmentStartTime, appointmentEndTime, appointmentDescription , destination)
SetSuggestedAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription , destination)
executeUpdate(insertA)
create(id,name, description, location,starttime, endtime, participants, dates,deadline)
ap: Appointment
ap
available(appointmentStartTime, appointmentEndTime)
available
res
res
add res to reqAnswer
addSuggestedAppointmnet
forward(reqAnswerS,resAnswerS)

CreatSuggestedAppointment / CreatPossibleAppointment – Remarks:

- reqAnswer represents a HTTPServletRequest object containing the required user input.
- resAnswer represents a HttpServletResponse object as the counterpart for the request.
- The state predicate AvailableApointment represents that the list of available Appointment is shown.
- ap refers to an object of class Appointment.
- available(…) checks whether the appointment is already available.
- res is a set of available appointments that fit to the selection criteria. The set will be added to the request object to be processed by the server.
- 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.

queryAP :

```
SELECT * FROM Appointment
WHERE (id=”aid”)
```

insertA :

```
INSERT INTO  Appointment
  ( id,name, description,location,participants,startTime,endTime)  values

(“aid”,“name”,”description”,”location”,0,”startTime”,”endTime”)
```

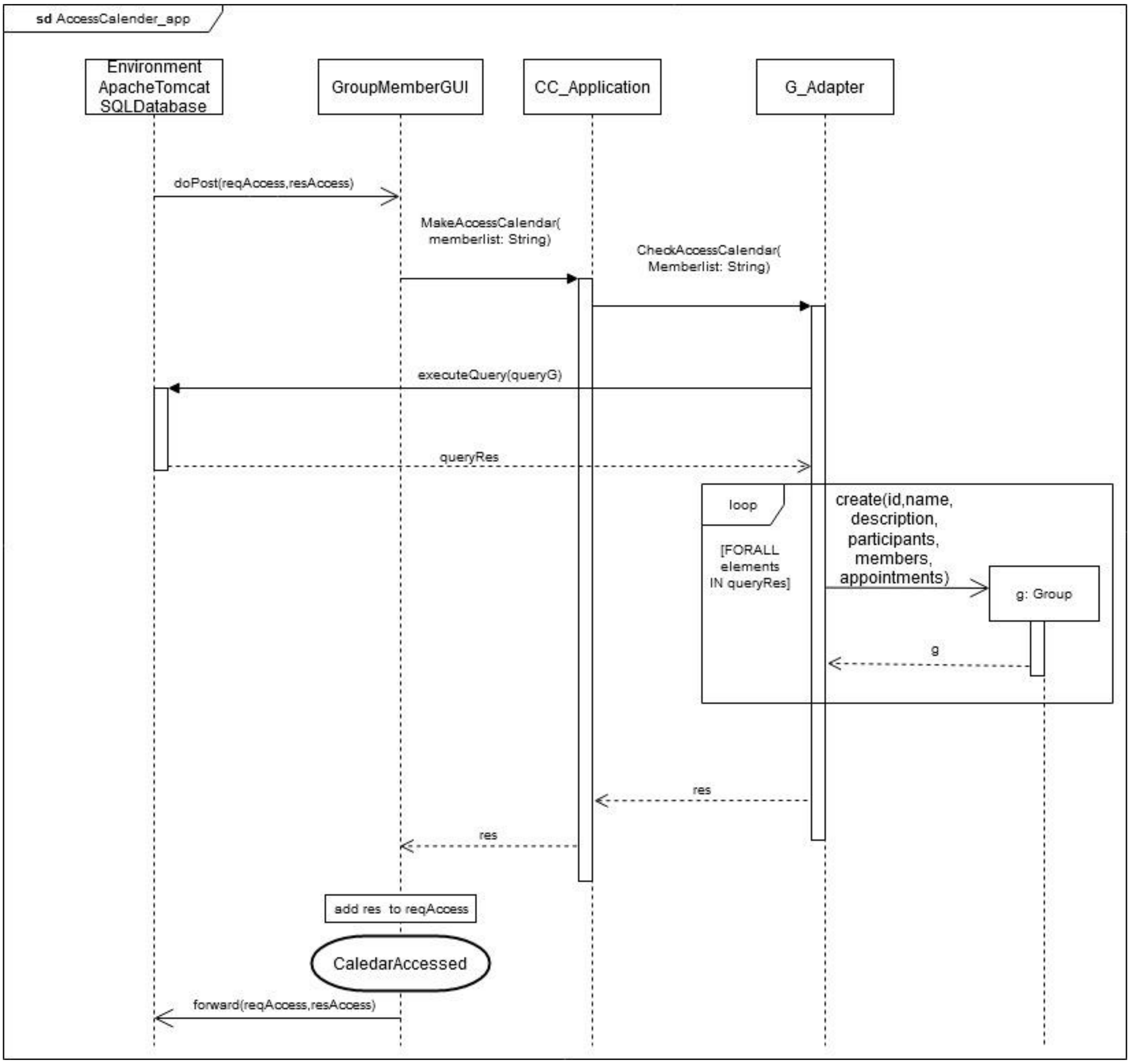
Inter-component interaction

MakeAccessCalendar– Operation specification :

OCL constraint:

```
context CC_Access :: MakeAccessCalendar (memberslist: String)
Pre: true
Post: let res: Set (Group) = g ->select (a: group | a.members = memberslist) ->asSet()
      in
          MemberPage^ShowAccessCalendarRequest(res)
```

sdAccessCalendar_app:


sd AccessCalender_app
Environment ApacheTomcat SQLDatabase
GroupMemberGUI
CC_Application
G_Adapter
doPost(reqAccess,resAccess)
MakeAccessCalendar(memberlist: String)
CheckAccessCalendar(Memberlist: String)
executeQuery(queryG)
queryRes
loop
[FORALL elements IN queryRes]
create(id,name, description, participants, members, appointments)
g: Group
g
res
res
add res to reqAccess
CaledarAccessed
forward(reqAccess,resAccess)

MakeAccessCalendar- Remarks:

- reqAccess represents a HTTPServletRequest object containing the required user input.
- resAccess represents a HttpServletResponse object as the counterpart for the request.
- g refers to an object of class Group.
- res is a list of members provided by the lexical domain Group to check the existence of a user in the group.
- forward(…) sends the request and response back to the server to generate the HTML webpage.

queryG :

SELECT * FROM Group
WHERE (members ="name")

- 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.

Inter-component interaction

DeadlineOfRequest - Operation specification :

OCL constraint:

context CC_Finalize :: DeadlineOfRequest ()

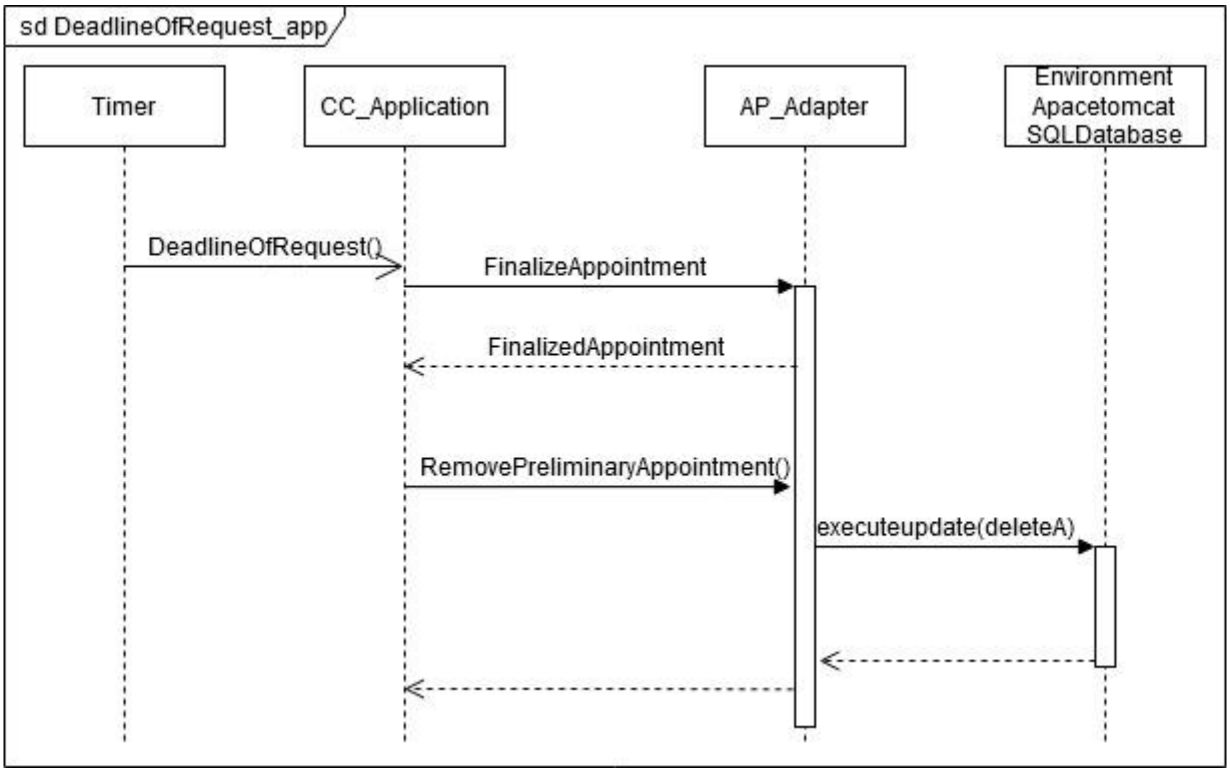
Pre: true

Post: let FinalizedAppointmentToBeDeleted: **set** (Appointment) = ap@pre ->select (a: Appointment | a.deadline.isEarlier(currentDate()) and a.participants = 0) ->asSet()

in

ap=ap@pre ->-(FinalizedAppointmentToBeDeleted)

sdDeadlineOfRequest_app:



DeadlineOfRequest - Remarks:

- ap refers to an object of class Appointment.
- 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.

- TIMESTAMPADD(interval, value, timestamp) is a function provided by SQL to add di_erent intervals to a given timestamp.
- SQL_TSI_DAY is a constant that defines that an interval of days is added

deleteA :

DELETE FROM Appointment **WHERE** (Participants = 0 AND (TIMESTAMPADD(SQL_TSI_Day, 0, deadline) < CURRENT_TIMESTAMP))

D2: Validation:

1. The sequence diagrams must be consistent with the behavior described in Step A3: Abstract software specification and in Step A6: Software lifecycle :

- Consistency of sdAddNewAppointment_app and sdAdd

Message in D2	**Corresponding message in A3**
doPost(reqAdd,resAdd)	refines RequestNewAppointment
CreateNewAppointment(…)	CreateNewAppointment
executeUpdate(InsertA)	refines SetNewAppointment
Available(…)	refines SetNewAppointment
Forward(reqAdd,resAdd)	Refines AppointmentsRepresntation

- Consistency of sdAnswerAppointment_app and sdAnswer

Message in D2	**Corresponding message in A3**
doPost(reqAnswer,resAnswer)	refines SelectPossibleAppointment
	refinesSuggestNewAppointment
CreatePossibleAppointment(…)	CreatePossibleAppointment
CreateSuggestedAppointment(…)	CreateSuggestedAppointment
executeQuery(queryAP)	refines SetPossibleAppointment
executeUpdate(InsertA)	refines SetSuggestedAppointment
Available(…)	refines SetPossibleAppointment
	refines SetSuggestedAppointment
Forward(reqAnswer,resAnswer)	Refines AppointmentsRepresntation

- Consistency of sdAccessCalendar_app and sdAccess

Message in D2	**Corresponding message in A3**
doPost(reqAccess,resAccess)	refines RequestAccessCalendar
makeAccessCalendar(…)	makeAccessCalendar
CheckAccessCalendar(…)	CheckAccessCalendar

executeUpdate(queryG)	refines CheckAccessCalendar
Forward(reqAccess,resAccess)	Refines GroupRepresntation

- Consistency of sdDeadlineOfRequest_app and sdFinalize

Message in D2	**Corresponding message in A3**
DeadLineOfRequest()	DeadLineOfRequest
RemovePreliminaryAppointment()	RemovePreliminaryAppointment
executeUpdate(deleteA)	Refines RemovePreliminaryAppointment

- Consistency with life-cycle:
  - Add , Answer, access and Finalize :
    sdAddNewAppointmnet_app, sdAnswerAppointment_app, sdAccessCalender_app and sdDeadlineOfRequest_app can be executed an arabitrary number of times without a precondition.
  - Add | Answer | access :
    sdAddNewAppointmnet_app, sdAnswerAppointment_app and sdAccessCalender_app can be executed alternatively as described in life-cycle.
  - (Add | Answer | access)* || Finalize:
    sdAddNewAppointmnet_app or sdAnswerAppointment_app or sdAccessCalender_app can be executed concurrently with sdDeadlineOfRequest_app without unwanted side-effects.

2. The sequence diagrams must realize the operation described in step A5:Operations and data specification :

- CreateNewApointment(…) is realized in sdAddNewAppointmnet_app :
  - Precondition does not have to be established, because it is true.
  - Postcondition is established, because a corresponding appointment is add using insertA and then AP_Adapter will check whether are available using the operation available(…).CC_Application forwards the result to GroupMemberGUI, that realizes MemberPage^ShowChoosedAppointmentRequest(res).
- CreatePossibleAppointment(…) is realized in sdAnswerAppointment_app:
  - Precondition does not have to be established, because it is true.
  - Postcondition CC_Application delegates the message to AP_Adapter using the SQL command queryAP, AP_Adapter selects all Appointment with a correct starTime and endTime for the given parameters. AP_Adapter will check whether are available appoinmtents using the operation available(…).CC_Application forwards the result to GroupMemberGUI, that realizes MemberPage^ShowChoosedPossibleAppointment(res).
- CreateSuggestedAppointment(…) is realized in sdAnswerAppointment_app:

    - Precondition does not have to be established, because it is true.
    - Postcondition is established, because a corresponding appointment is add using insertA and then AP_Adapter will check whether are available using the operation available(…).CC_Application forwards the result to GroupMemberGUI, that realizes MemberPage ^ ShowChoosedPossibleAppointment(res).
- MakeAccessCalendar(…) is realized in sdAccessCalender_app:
    - Precondition does not have to be established, because it is true.
    - Postcondition is established, because the requested group is selected using the SQL command queryG. CC_Application forwards the result to GroupMemberGUI, that realizes MemberPage^ShowAccessCalendarRequest(res).
- DeadlineOfRequest(…) is realized in sdDeadlineOfRequest_app:
    - Precondition does not have to be established, because it is true.
    - Postcondition is established, because deleteA removes all not participated appointments that are passed their deadline.

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

Interface	**Message**	**Used in sequence diagram**
GMCmds	CreateNewAppointment CreatePossibleAppointment CreateSuggestedAppointment makeAccessCalendar	sdAddNewAppointmnet_app sdAnswerAppointment_app sdAccessCalender_app
IAppointment	SetNewAppointment SetPossibleAppointment SetSuggestedAppointment RemovePreliminaryAppointment	sdAddNewAppointmnet_app sdAnswerAppointment_app sdDeadlineOfRequest_app
IGroup	CheckAccessCalendar	sdAccessCalender_app
ITimer	DeadlineOfRequest	sdDeadlineOfRequest_app

4. the directions of messages must be consistent with the required and provided interfaces of step D1: Software architecture.

Interface	**Provided by**	**Required by**
Message	**Recipient**	**Sender**
GMCmds	CC_Application	GroupMemberGUI
CreateNewAppointment	CC_Application	GroupMemberGUI

CreatePossibleAppointment	CC_Application	GroupMemberGUI
CreateSuggestedAppointment	CC_Application	GroupMemberGUI
makeAccessCalendar	CC_Application	GroupMemberGUI

IAppointment	AP_Adapter	CC_Application
SetNewAppointment	Ap_Adapter	CC_Application
SetPossibleAppointment	Ap_Adapter	CC_Application
SetSuggestedAppointment	Ap_Adapter	CC_Application
RemovePreliminaryAppointment	Ap_Adapter	CC_Application

IGroup	G_Adapter	CC_Application
CheckAccessCalendar	G_Adapter	CC_Application

ITimer	CC_Application	Timer
DeadlineOfRequest	CC_Application	Timer

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

Component	**Connected components in architecture**	**Connected components in sequence diagrams**
CC_Application	AP_Adapter, G_Adapter, Timer, GroupMemberGUI	AP_Adapter, G_Adapter, Timer, GroupMemberGUI
AP_Adapter	CC_Application, Environment	CC_Application, Environment
G_Adapter	CC_Application, Environment	CC_Application, Environment
GroupMemberGUI	CC_Application, Environment	CC_Application, Environment
Timer	CC_Application	CC_Application

D3

Intra-Component Interaction:

For our Project the intra-Components Interaction is not necessary because we don't have any complicated components, that we have to disassemble in sub-components.

Beside that we didn't introduce any new component in A5.
Therefore, There is no need to specify an intra-component interaction for neither of our Sequence Diagrams.

Complete component or class behavior I

Procedure

Check whether a state machine is necessary:

- The component Timer is a re-used components. It is not necessary to create state machines for it.
- The component CC_Application: there is no refinement of this component in Step D3: Intra-Component Interaction; continue looking at Step D2: Inter-Component Interaction. 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 GroupMember. It is not necessary to create a state machine for this component.
- The component AP Adapter: no refinement exists in Step D3: Intra-Component Interaction; continue with looking at Step D2: Inter-Component Interaction. Furthermore, it is not necessary to create state machines for the components Appointment as the data base with its corresponding DBMS handles the states and state changes.
- The component G Adapter: no refinement exists in Step D3: Intra-Component Interaction; continue with looking at Step D2: Inter-Component Interaction. Furthermore, it is not necessary to create state machines for the components Group as the data base with its corresponding DBMS handles the states and state changes.
- The component GroupMemberGUI: no refinement exists in Step D3: Intra-Component Interaction; continue with looking at Step D2: Inter-Component Interaction. There are more than two states. Therefore, a state machine is required.

Complete component or class behavior

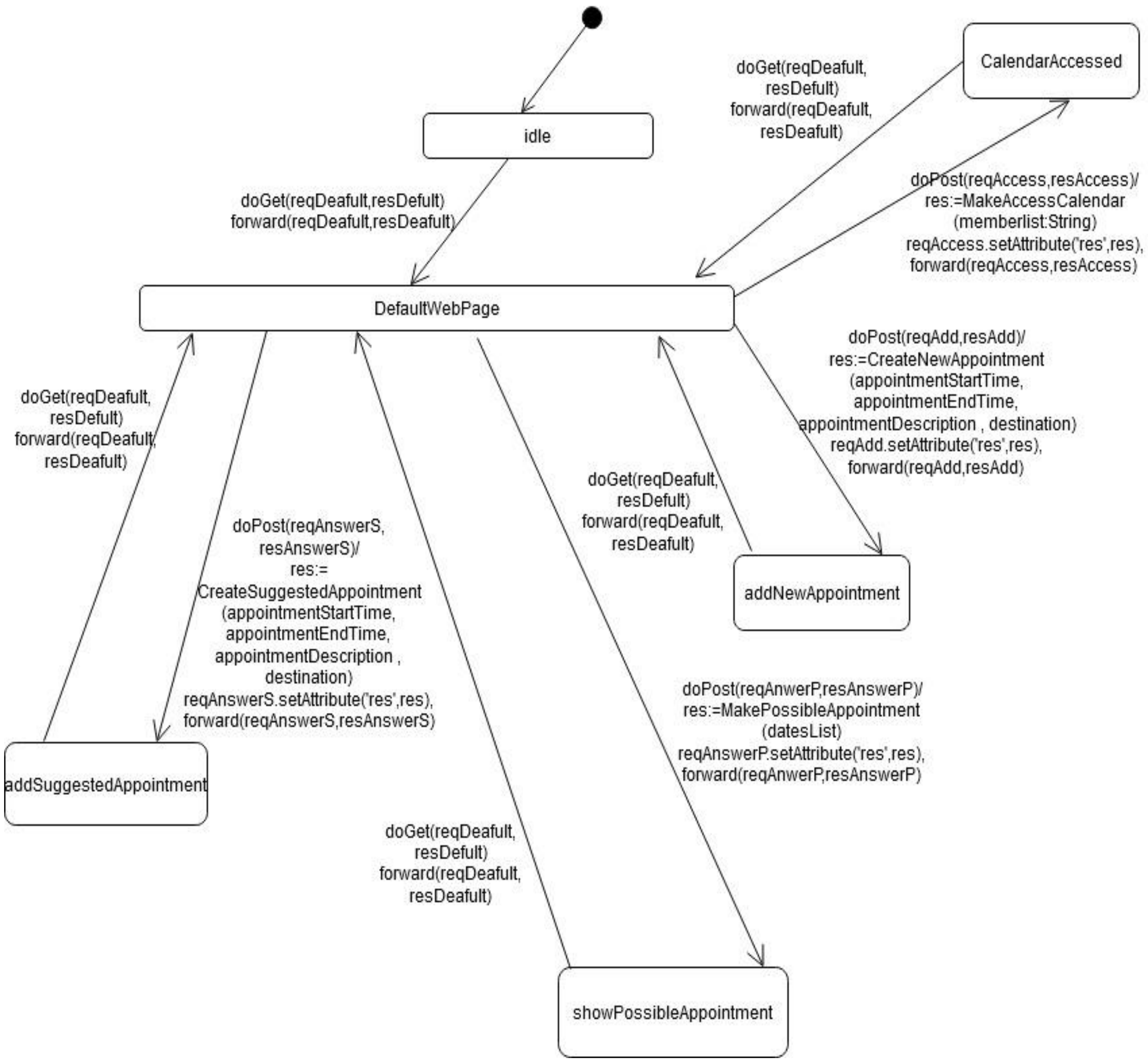
Procedure 2

For the necessary state machines, perform the following according to the corresponding sequence diagrams: Considering GroupMemberGUI:

- messages doGet(...) and doPost(...) must be triggers at transitions.
- messages CreatNewAppointment(...), CreatePossibleAppointment(...), CreateSuggestedAppointment(…), MakeAccessCalendar(…) and forward(...) must be output signals or actions state predicates avilableAppointment must be states.
- Idle and ShowDefaultWebpage are new states required to represent the functionalities of the web application

Complete component or class behavior

State Machine GroupMemberGUI



idle
CalendarAccessed
DefaultWebPage
addNewAppointment
addSuggestedAppointment
showPossibleAppointment
doGet(reqDeafult,resDefult) forward(reqDeafult,resDeafult)
doGet(reqDeafult, resDefult) forward(reqDeafult, resDeafult)
doPost(reqAccess,resAccess)/ res:=MakeAccessCalendar (memberlist:String) reqAccess.setAttribute('res',res), forward(reqAccess,resAccess)
doPost(reqAdd,resAdd)/ res:=CreateNewAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription , destination) reqAdd.setAttribute('res',res), forward(reqAdd,resAdd)
doGet(reqDeafult, resDefult) forward(reqDeafult, resDeafult)
doGet(reqDeafult, resDefult) forward(reqDeafult, resDeafult)
doPost(reqAnswerS, resAnswerS)/ res:= CreateSuggestedAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription , destination) reqAnswerS.setAttribute('res',res), forward(reqAnswerS,resAnswerS)
doPost(reqAnwerP,resAnswerP)/ res:=MakePossibleAppointment (datesList) reqAnswerP.setAttribute('res',res), forward(reqAnwerP,resAnswerP)
doGet(reqDeafult, resDefult) forward(reqDeafult, resDeafult)

Complete component or class behavior

Remarks

We add the following additional transitions to model the complete behavior 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.

The Validation (D4):

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

ComponentGroupMemberGUI (Add)					
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	addNewAppointment	doPost(reqAdd,resAdd)	doPost (reqAdd,resAdd)	res:= CreateNewAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription , destination) reqAdd.setAttribute ('res',res), forward(reqAdd,resAdd)	CreateNewAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription , destination) forward(reqAdd,resAdd)
DefaultWebPage	ShowPossibleAppointment	doPost(reqAnswerP,resAnswerP)	doPost(reqAnswerP,resAnswerP)	res:=MakePossibleAppointment (datesList) reqAnswerP.setAttribute('res',res), forward(reqAnwerP,resAnswerP)	MakePossibleAppointment (datesList), forward(reqAnwerP,resAnswerP)
DefaultWebPage	addSuggestedAppointment	doPost(reqAnswerS,resAnswerS)	doPost(reqAnswerS,resAnswerS)	res:= CreateSuggestedAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription, destination) reqAnswerS.setAttribute('res',res), forward(reqAnswerS,resAnswerS)	CreateSuggestedAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription , destination) forward(reqAnswerS, resAnswerS)
DefaultWebPage	CalendarAccessed	doPost(reqAccess,resAccess)	doPost(reqAccess,resAccess)	res:=MakeAccessCalendar (memberlist:String) reqAccess.setAttribute('res',res), forward(reqAccess,resAccess)	MakeAccessCalendar (memberlist:String) reqAccess.setAttribute('res',res), forward(reqAccess,resAccess)
addNewAppointment	DefaultWebPage	doGet(reqDefault, resDefault)	-	forward(reqDefault,resDefault)	-
showPossibleAppointment	DefaultWebPage	doGet(reqDefault, resDefault)	-	forward(reqDefault,resDefault)	-
addSuggestedAppointment	DefaultWebPage	doGet(reqDefault, resDefault)	-	forward(reqDefault,resDefault)	-
CalendarAccessed	DefaultWebPage	doGet(reqDefault, resDefault)	-	forward(reqDefault,resDefault)	-

- The state machines are consistent with the life-cycle model of Step A6: Software lifecycle.
- All states are covered by a life-cycle.

Component MemberPageGUI	
$LC_{\text{group member}}$ = (Add | Access | Answer) *	
State	**Covered by Life Cycle Part**
Init	Add
Init	Answer
Init	Access
DefaultWebPage	Add
DefaultWebPage	Answer
DefaultWebPage	Access
addNewAppointment	Add
showPossibleAppointment	Answer
addSuggestedAppointment	Answer
CalendarAccessed	Access

- All transitions are covered by a life-cycle.

Component MemberPageGUI				
$LC_{\text{group member}}$ = (Add | Access | Answer) *				
Source State	**Target State**	**Input Signal**	**Output Signal**	**Life cycle part**
Init	DefaultWebPage	doGet(reqDefault,resDefault)	forward(reqDefault,resDefault)	(Add |
Init	DefaultWebPage	doGet(reqDefault,resDefault)	forward(reqDefault,resDefault)	Access |
Init	DefaultWebPage	doGet(reqDefault,resDefault)	forward(reqDefault,resDefault)	Answer)
DefaultWebPage	addNewAppointment	doPost(reqAdd,resAdd)	res:= CreateNewAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription , destination)	(Add |

			reqAdd.setAttribute ('res',res), forward(reqAdd,resAdd)	
DefaultWebPage	CalendarAccessed	doPost(reqAccess,resAccess)	res:=MakeAccessCalendar (memberlist:MemberData) reqAccess.setAttribute('res',res), forward(reqAccess,resAccess)	Access |
DefaultWebPage	showPossibleAppointment	doPost(reqAnswer,resAnswer)	res:=MakePossibleAppointment (datesList) reqAnswerP.setAttribute('res',res), forward(reqAnwerP,resAnswerP)	Answer)
DefaultWebPage	addSuggestedAppointment	doPost(reqAnswerS,resAnswerS)	res:= CreateSuggestedAppointment (appointmentStartTime, appointmentEndTime, appointmentDescription, destination) reqAnswerS.setAttribute('res',res), forward(reqAnswerS,resAnswerS)	Answer)
addNewAppointment	DefaultWebPage	doGet(reqDefault,resDefault)	forward(reqDefault,resDefault)	(Add)*
showPossibleAppointment	DefaultWebPage	doGet(reqDefault,resDefault)	forward(reqDefault,resDefault)	(Answer)*
addSuggestedAppointment	DefaultWebPage	doGet(reqDefault,resDefault)	forward(reqDefault,resDefault)	(Answer)*
CalendarAccessed	DefaultWebPage	doGet(reqDefault,resDefault)	forward(reqDefault,resDefault)	(Access)*

Glossary :

Name	Kind	Description	Source
A			
A		Abbreviation for Administrator	CD
AccessCalendarChecked	phenomenon	Allow/deny access to the Calendar	CD, PdAccess, sdAccess
Administrator	Biddable domain	The admin of the group	CD
AddAppointment	phenomenon	Add appointment to the calendar	CD
addNewAppointment	State	Represents that a new appointment has been added	sdAddNewAppointment_app State Machine GuestGUI,
addSuggestedApointment	State	Represents that a new suggested appointment has been added	sdAnswerAppointment_app State Machine GuestGUI
AllowAccessCalendar	phenomenon	Allow the user to access the Calendar	CD
AllowParticipation	phenomenon	Allow the user to participate with the new appointment	CD
AP		Abbreviation for Appointment	CD, PdAdd, PdAnswer, PdFinalize
APAdapterPort	port	Interaction points of AP_Adapter with the environment.	subArchAdd subArchAnswer globalArch
APIProvider			
Appointment	Lexical domain	Appointment created by a user or a member of the group	CD, PdAdd, PdAnswer, PdFinalize, sdAdd, sdAnswer, sdFinalize, Class model
Appointments	Attribute	An appointment from the app	sdAccessCalender_app
APPort	port	Interaction points of IAppointment, AP_Adapter with the environment.	subArchAdd subArchAnswer globalArch
APPort_I	port	Interaction points of IAppointment, AP_Adapter with the environment.	subArchAdd subArchAnswer globalArch
AssignAdministrator	phenomenon	Assign a new admin	CD
Available()	Message, auxiliary function	Is the appointment available,	sdAddNewAppointment_app, sdAnswerAppointment_app,sd
ApacheTomcat	Connection domain	An Open Source JSP and Servlet Container from the Apache Foundation.	TCD, sdAddNewAppointment_app sdAnswerAppointment_app sdAccessCalender_app sdDeadlineOfRequest_app
AP_Adabpter	component	responsible to create and maintain tables for all persistent classes	subArchAdd subArchAnswer globalArch, sdAddNewAppointment_app sdAnswerAppointment_app sdDeadlineOfRequest_app
AppointmentStartTime	State predicate	The time when the appointment will start	Class model, Sd AddNewAppointment_app State Machine GuestGUI

			sdAnswerAppointment_app
AppointmentEndTime	State predicate	The time when the appointment will end	Class model, Sd AddNewAppointment_app sdAnswerAppointment_app State Machine GuestGUI
AppointmentDescription	State predicate	The description of the appointment	Class model, Sd AddNewAppointment_app State Machine GuestGUI
Available Appointments	State predicate	Represents that an appointment which is available is shown	SdAddNewAppointment_app, sdAnswerAppointment_app sd add,sd Asnwer
B			
C			
CC		Abbreviation for Collaborative Calendar	CD
CCA		Abbreviation for CC_Add	PdAdd
CCF		Abbreviation for CC_Finalize	PdFinalize
CCAC		Abbreviation for CC_Access	PdAccess
CCAN		Abbreviation for CC_Answer	PdAnswer
CC_Access	Machine	Collaborative Calendar app to display calendar to the Group members	PdAccess, sdAccess, Class model
CC_Add	Machine	Collaborative Calendar app to add new appointment	PdAdd, sdAdd, Class model
CC_Answer	Machine	Collaborative Calendar app to answer new appointment	PdAnswer, sdAnswer, Class model
CC_Application	component	Collaborative Calendar Application	subArchAdd subArchAnswer subArchAccess subArchFinalize globalArch, sdAddNewAppointment_app sdAnswerAppointment_app sdAccessCalender_app sdDeadlineOfRequest_app
CC_Finalize	Machine	Collaborative Calendar app to add fix appointment	PdFinalize, sdFinalize, Class model
CalendarAccessed	State predicate	Represents that the Calendar has been accessed from a group member	sdAccessCalendar_app, sd Access
ChangeRights	phenomenon	The admin can change the rights of the group members	CD
CheckAccessCalender	phenomenon	Check whether the user can access the group calendar	CD, PdAccess, sdAccess,sdAccess,sdAccessCalender_app
CheckParticipation	phenomenon	Check whether the user can Participate with new appointment request	CD
Collaborative Calendar	Machine,component,layered_architecture	Collaborative Calendar App	CD
Create	Message,auxiliary function	The users create a new group	sdAccess,sdAccessCalender_app,

			sdAddNewAppointment_app sdAnswerAppointment_app sdDeadlineOfRequest_app
CreatGroup	phenomenon	The user create new group	CD
CreatPossibleAppointment	Phenomenon, auxiliary function	Possible appointments have been created	PdAnswer, sdAnswer,sdAnswerAppointment_app,
CreatNewAppointment	Phenomenon, auxiliary function	New appointment has been created	PdAdd, sdAdd,Class model
CreaterAsAdministrator	phenomenon	Make the group creator admin for the group	CD
CreateSuggestedAppointment	Phenomenon, auxiliary function	Create the appointments which were suggested	sdAnswerAppointment_app
D			
Day	Attribute	The day of the Appointment	Class Model
dates	attribute	The date of the possible Appointment	Class model,Sd AddNewAppointment_app, sdAddNewAppointment_app sdAnswerAppointment_app
deadline	state predicate	Deadline is reached	sdFinilize
DeadlineOfRequest	State predication, auxiliary function	message for the deadline of the participate request	sdFinalize, Class model,sdDeadlineOfRequest_app
DedaultWebPage	state	Indicates that the server waits for incoming repuest	State Machine GuestGUI
description	attribute	The description of the Appointment	Class model, sdAddNewAppointment_app sdAnswerAppointment_app sdAccessCalender_app
doGet	phenomenon	Information being got	TCD State Machine GuestGUI
doPost	phenomenon ,message,auxiliary function	Information being posted	TCD,Sd State Machine GuestGUI AddNewAppointment_app,sdAccessCalender_app
destination	attribute	The destination of the appointment	Sd AddNewAppointment_app sdAnswerAppointment_app State Machine GuestGUI
E			
endTime	attribute	The time when the Appointment is end	Class model, sdAddNewAppointment_app sdAnswerAppointment_app
executeQuery	phenomenon	The query is executed by the machine	TCD
	message	Java API function to send an SQL up date command to a MySQL database.	sdAnswerAppointment_app , sdAccessCalender_app,
executeUpdate	phenomenon	The update is executed by the machine	TCD
	message	Java API function to send an SQL up date command to a	sdAddNewAppointment_app sdAnswerAppointment_app sdDeadlineOfRequest_app

		MySQL database.	
Environmen		Represent the environment of the app	sdAddNewAppointment_app sdAnswerAppointment_app sdAccessCalender_app sdDeadlineOfRequest_app
F			
FinalizeAppointment	phenomenon	Add fixed appointment to the Calendar	CD, PdFinalise, sdFinalize
FinalizedAppointment	phenomenon	Fixed appointment added to the Calendar	CD, PdFinalise
forward	phenomenon	Command go forward, sends the request and response back to the server to generate the HTML webpage.	TCD,sdAddNewAppointment, sdAnswerAppointment_app sdAccessCalender_app State Machine GuestGUI
G			
G		Abbreviation for Group	CD, PdAccess, PdFinalise
G_Adapter	component	responsible to create and maintain tables for all persistent classes	subArchAccess subArchFinalize globalArch, sdAccessCalender_app
GAdapterPort	port	Interaction points of G_Adapter with the environment.	subArchAdd subArchAnswer subArchAccess globalArch
GM		Abbreviation for Group Member	CD, PdAdd, PdAnswer
GMCmds	interface	Skeleton from problem diagrams and messages in sd. Group member's commands.	subArchAdd subArchAnswer subArchAccess globalArch
GMCmdsPort	port	Interaction points of CC_Application GMCmds GroupMemberGUI with the environment.	subArchAdd subArchAnswer subArchAccess globalArch
GMCmdsPort_I	port	Interaction points of GMCmds GroupMemberGUI with the environment.	subArchAdd subArchAnswer subArchAccess globalArch
GPort	port	Interaction points of CC_Application IGroup G_Adapter with the environment.	subArchAccess globalArch
GPort_I	port	Interaction points of IGroup G_Adapter with the environment.	subArchAccess globalArch
Group	Lexical domain	Data base for the members of the group	CD, PdAccess, PdFinalise, sdAccess, sdFinalize, Class model
Group Member	Biddable domain	A Person may join the Group	CD, PdAdd, PdAnswer, PdAccess, sdAdd, sdAnswer, sdAccess
GroupMemberGUI	component	web interface for users	subArchAdd globalArch, sdAddNewAppointment, sdAnswerAppointment_app

			sdAccessCalender_app
GroupMemberPort	port	Interaction points of GroupMemberGUI with the environment.	subArchAdd globalArch
H			
hour	attribute	The hour of the Appointment	Class model
I			
IAppointment	interface	Used for internal operations in subproblmes	subArchAdd subArchFinalize globalArch
idle	state	Indicates the starting page.	State Machine GuestGUI
IGroup	interface	Used for internal operations in subproblmes	subArchAccess subArchFinalize globalArch
id	Attribute	The id of the Appointment	Class model
		The Id of the Group	sdAccess,sdAccessCalender_app
		The id of appointments	sdAddNewAppointmentCalender_app
inviteUser	phenomenon	admin invite user	CD
isEarlier()	auxiliary function	Is the user earlier or not	Class model
ITimer	interface	Used to trigger the internal operation "DeadlineOfRequest" periodically	subArchFinalize globalArch
java.servlet.http.HttpServlet	interface	Interact with websites with commands.	subArchAdd subArchAnswer subArchAccess subArchFinalize globalArch
Java.sql.statement	interface	Help with SQL language	subArchAdd subArchAnswer subArchAccess subArchFinalize globalArch
JoinGroup	phenomenon	User can join the group	CD
K			
L			
$LC_{\text{group member}}$	life-cycle	life-cycle for one group member	LC
$LC_{\text{collaborative calendar}}$	life-cycle	Combined life-cycle (all group members and the internal operation)	LC
LogIn	phenomenon	User can log in	CD
LogOut	phenomenon	User can log out	CD
location	attribute	The location of the Appointment	Class model, sdAddNewAppointment_app sdAnswerAppointment_app

M			
MakeAccessCalendar	Phenomenon,message,auxiliary function	Access calendar has been requested	PdAccess, sdAccess,sdAccessCalender_app
MP		Abbreviation for MemberPage	
MarkPreliminaryAppointment	Phenomenon	Mark preliminary appointments in the Calendar	CD
Members	Attribute	Member of the group	sdAccessCalender_app
MemberPage	Domain	The connection domain between the GroupMember and the machine. Forwarding the inputs of the GroupMember to the machine and the outputs of the machine to the guest.	PdAdd, PdAnswer, PdAccess, sdAdd, sdAnswer, sdAccess, Class model
MemberWebBrowser	connectionDomain	A browser that member can view their event.	TCD
MemberData	dataType	The data of the member	Class model,sdAccess,sdAccessCalender_app State Machine GuestGUI
Memberlist	dataType	The list of member	sdAccess,sdAccessCalender_app State Machine GuestGUI
minute	attribute	The minute of the Appointment	Class model
month	attribute	The month of the Appointment	Class model
N			
name	attribute	The name of the user	Class model, sdAddNewAppointment_app sdAnswerAppointment_app sdAccessCalender_app
		The name of the appointment	
NewAppointment	phenomenon	New appointment request added to the Database	CD, PdAdd, sdAdd
NewAdministratorAssigned	phenomenon	New administrator assigned to the group in the database	CD
NewUserInvited	phenomenon	New user invited to the group in the database	CD
NewUserRegistered	phenomenon	New user registered to the app in the database	CD
O			
P			
P		Abbreviation for Person	CD
Person	Biddable Domain	Person may use the app	CD
Participate	phenomenon	User can participate	CD
participants	Attribute	The data of the participant	Class model, sdAddNewAppointment_app sdAnswerAppointment_app sdAccessCalender_app
ParticipationChecked	phenomenon	Allow/deny the Participation of the User	CD

PossibleAppiontment	phenomenon	Possible appointments added to the Database	CD, PdAnswer, sdAnswer
Q			
R			
Register	phenomenon	User can register	CD
RemovePreliminary Appointment	Phenomenon,message,auxiliary function	Remove preliminary appointments from the Calendar	CD, PdFinalise, sdFinalize,sdDeadlinOfRequest sdDeadlineOfRequest_app
RequestAccessCalendar	phenomenon	User can access the Calendar	CD, PdAccess, sdAccess
RequestNewAppointment	Phenomenon, auxiliary function	Group member can request new appointment	CD, PdAdd, sdAdd,Class model
RightsChanged	phenomenon	The group's rights changed in the database	CD
reqAdd	State predicate	represent HTTPServletRequest objects containing the required user input.	sdAddNewAppointment_app State Machine GuestGUI
resAdd	State predicate	represent HttpServletResponse objects as the counterpart for the request.	sdAddNewAppointment_app State Machine GuestGUI
reqAccess	State predicate	represent HTTPServletRequest objects containing the required user input.	sdAccessCalender_app State Machine GuestGUI
resAccess	State predicate	represent HttpServletResponse objects as the counterpart for the request.	sdAccessCalender_app State Machine GuestGUI
reqAnswer	State predicate	represent HTTPServletRequest objects containing the required user input.	sdAnswerAppointment_app State Machine GuestGUI
reqDeafult	State predicate	represent idle objects containing the required user input.	State Machine GuestGUI
resDeafult	State predicate	represent idle objects as the counterpart for the request.	State Machine GuestGUI
resAnswer	State predicate	represent HttpServletResponse objects as the counterpart for the request.	sdAnswerAppointment_app State Machine GuestGUI
S			
SelectPossibleAppointment	phenomenon	All Possible appointments has been selected	CD, PdAdd, PdAnswer, PdFinalise, sdAnswer
SetGroup	phenomenon	Create a new group in the data base	CD
SetNewAppointment	phenomenon	Add a new appointment to the Calendar	CD, PdAdd, sdAdd

SetPossibleAppointment	phenomenon	Add possible appointment to the Calendar	PdAnswer, sdAnswer
SetRequestDeadline	phenomenon	Approve a request's deadline	CD
SetSuggestedAppointment	phenomenon	Add suggested appointments to the Calendar	PdAnswer, sdAnswer,sdAnswerAppointment_app
ShowAccessCalendar Request	phenomenon	Display the Request for calendar access	PdAccess, sdAccess
ShowChoosedAppiontment Requeast	Phenomenon	Display the feasible Appointment request	PdAdd, sdAdd
ShowChoosedAppiontment Requeast	attribute	Display the feasible Appointment request	Class model
ShowChoosedPossible Appointment	phenomenon	Display feasible possible appointment	PdAnswer, sdAnswer
ShowCalendar	phenomenon	Display calendar to the Group member	PdAccess
ShowFinalizedAppointment	phenomenon	Display the fixed appointment	CD
ShowNewAppointment	phenomenon	Display the new appointment	CD
ShowPossibleAppointment	Phenomenon	Display possible appointment	CD
showPossibleAppointment	State	Represents that a possible appointment which is available is shown	sdAnswerAppointment_app State Machine GuestGUI
SQLDataBase	Causal domain	Representing the two Databases in the App(Group,Appointment)	TCD, sdAddNewAppointment_app sdAnswerAppointment_app sdAccessCalender_app sdDeadlineOfRequest_app
SuggestNewAppointment	phenomenon	Group member can suggest new appointment	CD, PdAnswer, sdAnswer
street	attribute	The street that the user lived at	Class model
startTime	attribute	The start time of the Appointment	Class model, sdAddNewAppointment_app sdAnswerAppointment_app
T			
Timer	Reused component	given component initiating the internal operation "DeadlineOfRequest "	subArchFinalize globalArch, sdDeadlineOfRequest_app
TimeData	dataType	The data of the Appointment time	Class model
TimerPort	port	Interaction points of CC_Application Timer with the environment.	globalArch
TimerPort_I	port	Interaction points of CC_Application Timer with the environment.	globalArch
town	attribute	The town that the user lived at	Class model
U			
U		Abbreviation for User	CD

Users	lexical domain	Data base for the people who may use Collaborative Calendar App	CD
UserJoined	phenomenon	Add a new user to the data base	CD
UserLogedIn	phenomenon	User logged in the app	CD
UserLogedOut	phenomenon	User logged out the app	CD
UserID	attribute	The ID of the user	Class model
V			
W			
X			
Y			
Year	attribute	The year of the Appointment	Class model
Z			