1 Revision History

Version	Date	Description
Inception draft	13/09/2012	Added Vision, Use Cases and UML, Glossary and Supplementary Requirements
Elaboration phase	27/09/2012	Added Domain Model, System Sequence Diagram and Operation Contracts
Further elaboration	04/10/2012	Added UML Package Diagrams, another Sequence Diagram and a Discussion of our software attributes, changed Supplementary Requirements(should to shall and performance specification), added revision history, added numbering and bullets to Operation Contracts, added titles to models, added package diagram, added use case diagram, added detailed sequence diagrams
Further modelling	11/10/2012	Added detailed sequence diagrams and a discussion of our software attributes. Added communication diagram and reflection
Scrum and GRASP	1/11/2012	Added backlog, burndown chart, GRASP principles and proposed changes to current design.

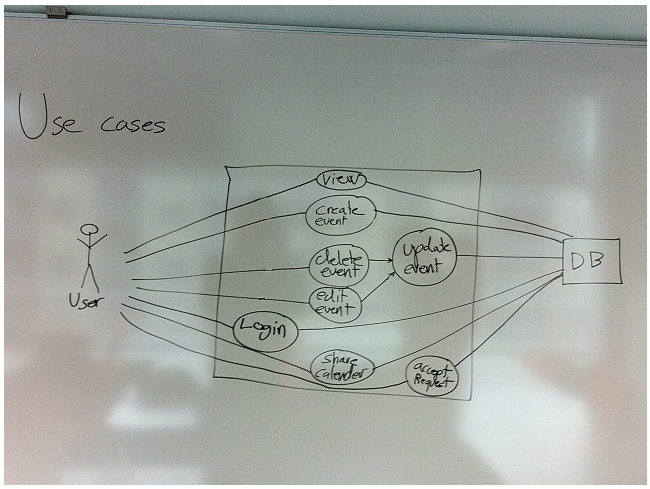
2 Glossery

Event Something predicted to happen at some point.

3 Vision

The purpose of this product is to make browsing, management and sharing easy for people with little or no knowledge about interaction with networks or computers. Performance shall be reasonable and use of the application shall not expose any security threats, unless prevention compromises usability or reliability.

4 User cases



Name View calender
Scope Calender client
Level User goal
Primary Actor User
Stakeholders and Interests User
Precondition

- User is connected to database.
- User has logged on.

Main Success Scenario User gets the desired information.
Extensions No database connection returns error

Name Create event
Scope Calender client
Level User goal
Primary Actor User
Stakeholders and Interests User
Precondition

- User is connected to database.
- User has logged on.

Postcondition

- An event has been created in the database.
- View is updated

Main Success Scenario User creates the desired event
Extensions No database connection returns error

Name Delete event
Scope Calender client
Level User goal
Primary Actor User
Stakeholders and Interests User
Precondition

- User is connected to database.
- User has logged on.
- User has chosen a single event in the users personal calender.

Postcondition

- the event has been deleted in the database.
- View is updated

Main Success Scenario User deletes the desired event
Extensions No database connection returns error

Name Edit event
Scope Calender client
Level User goal
Primary Actor User
Stakeholders and Interests User
Precondition

- User is connected to database.
- User has logged on.
- User has chosen a single event in the users personal calender.

Postcondition

- the event has been updated in the database.
- View is updated

Main Success Scenario User updates the desired event
Extensions No database connection returns error

Name User login
Scope Calender client
Level User goal
Primary Actor User
Stakeholders and Interests User
Precondition

- User is connected to database.

Postcondition The user has successfully logged on to the system
Main Success Scenario User logs on to the system
Extensions No database connection returns error Wrong e-mail address and/or paswords returns error and offers an opportunity to get the password sent by mail
Special Requirements

Name Share calender
Scope Calender client
Level User goal
Primary Actor User
Stakeholders and Interests User
Precondition User is connected to database. User is logged on
Postcondition The user has sent a share-invite to another user of the system
Main Success Scenario The user successfully sends a share-invite to another user
Extensions No database connection returns error The entered e-mail which the user wants to share the calender does not exist in the system. The user is presented with an error
Special Requirements

Name Accept request
Scope Calender client
Level User goal
Primary Actor User
Stakeholders and Interests User
Precondition User is connected to database. User is logged on User has received an invite to see another users calender
Postcondition The user has accepted the invite and is now able to see the shared calender. The view is updated.
Main Success Scenario The user has access to see the shared calender
Extensions No database connection returns error The user declines the invite, nothing happens
Special Requirements

5 Supplementary specification

The supplementary specification is structured using the FURPS+ model.

Functionality: The program shall support a minimum of functions.

- The functions shall support sharing of calenders, creation and modification of events and viewing.
- The functions shall be intuitively grouped.

Usability: The user shall be able to use the program with as little knowledge as possible, which raises the following requirements:

- The user shall consider a minimum of options performing the desired task. This may compromise both performance and functionality.
- Anyone beyond the age of 12 shall be able to understand the program terminology.

Reliability: The program shall be very reliable, even if it compromises security. Any error should result in recovery or crash, neither involving the user. Any crash or recovery must maintain the content of the database.

- The program shall be able to complete as many tasks as possible without internet connection.

Performance: The program shall have reasonable performance. This means that no operation shall take more than 2 seconds on a bandwidth with more than 2 mBit/s and a processor of more than 1 GHz.

Supportability: Functions shall be supported with a help page. The content of the help page should be limited because the intuitiveness and limited functionality makes it needless.

6 Discussion of software attributes

The system shall support a limited selection of functions. This decision makes it is easier to get an overview of the functionality for the novice user. This converge with our usability requirement, because users with limited knowledge will benefit from this. Security hinders usability, as it is harder to use a system that requires user authentication. The system has this trade-off because it is a minimum for supporting cloud computing. The program could involve the user in eventual errors, but this has been deselected to make it simpler. Only connectivity errors that users have a chance to solve will return an error. All of the decisions match our priority of making the system usable with as little knowledge as possible.

7 Reflection on Sequence diagrams and Communication diagrams

Sequence diagrams are in most cases easier to understand, as the vertical axis follows the chronological order. Communication diagrams relies on numbers for ordering of events, which takes longer to understand. Communication diagrams are however easier to edit and you don't need to think ahead when placing elements, like in a sequence diagrams. This makes communication diagrams more suited for development, and sequence diagrams for communication of finished design.

8 Operation Contracts

OC 1
Operation: newEvent(time:Date, description:String)

References: User Cases: Create new event

Pre Conditions:

- User must be connected to the server
- must be logged on to the system

Post Conditions:

- An Event instance event was created (instance creation)
- event.ID is uniqe (Primary key established)
- event is associated with a Calendar, based on the users ID (association formed)
- event.description becomes description (attribute modefication)

OC 2
Operation: updateEvent(event:Event, command:String)

References: User Cases: Edit event, Delete event

Pre Conditions:

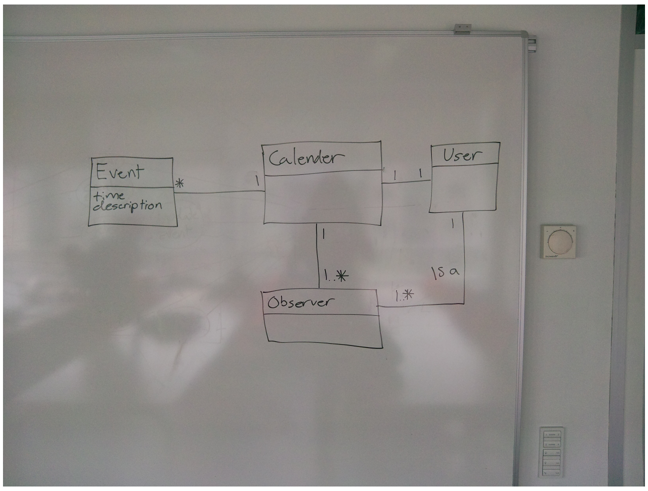
- User must be connected to the server
- User must be logged on to the system
- An exitsing Event must be selected

Post Conditions:

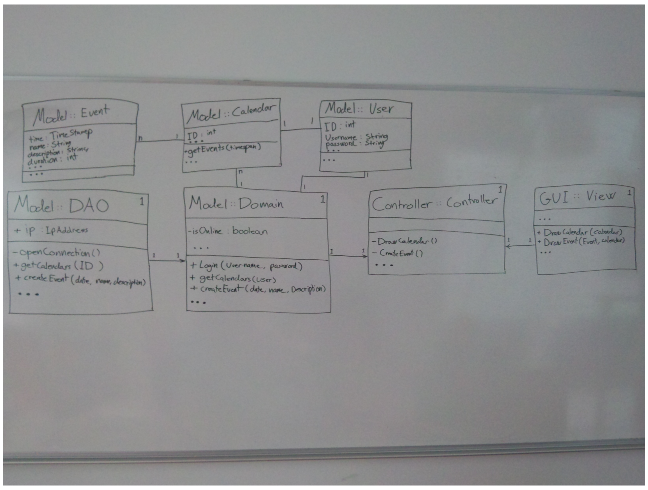
- The incomming command has been executed on the target Event (instance modification)

9 Models

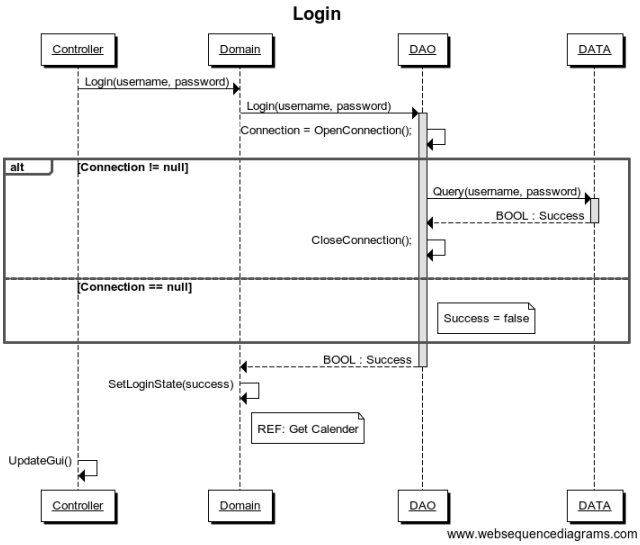
Domain Model

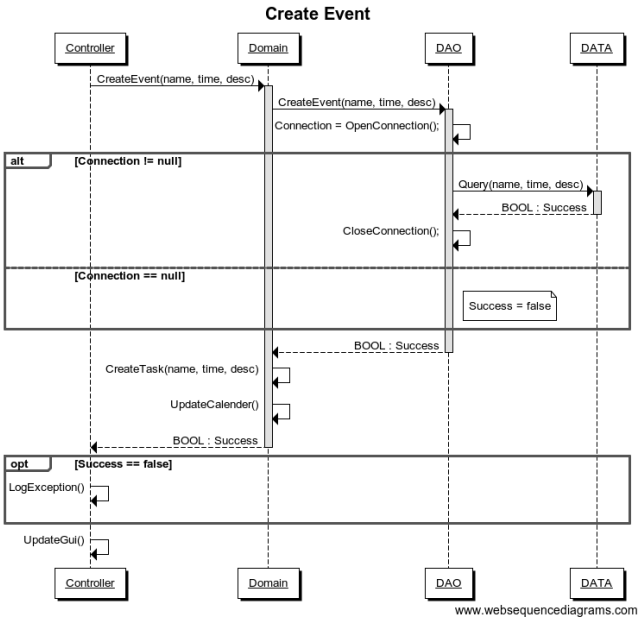


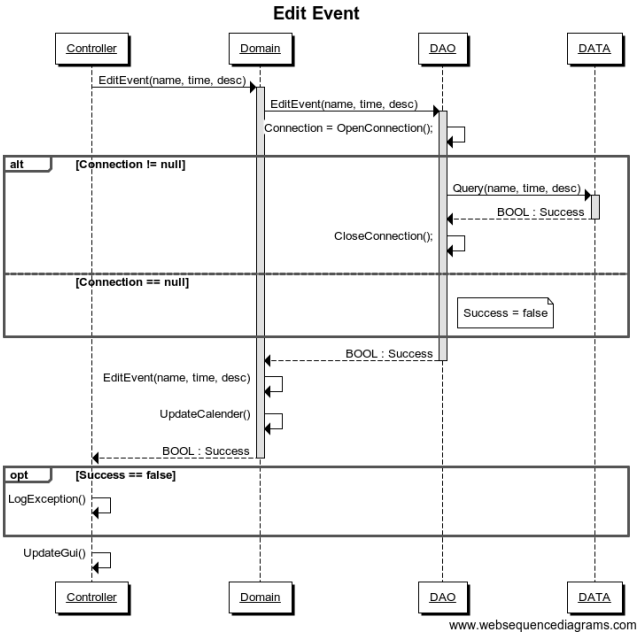
Design Class Diagrams



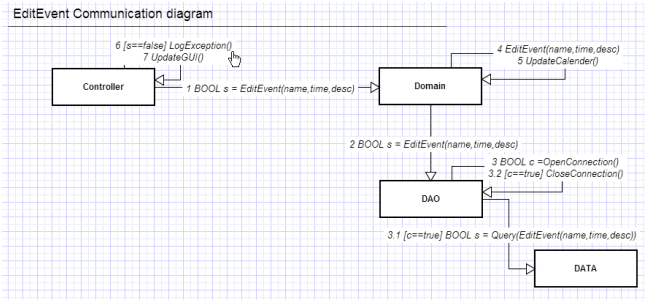
Sequence diagrams


Login
Controller
Domain
DAO
DATA
Login(username, password)
Login(username, password)
Connection = OpenConnection();
alt
[Connection != null]
Query(username, password)
BOOL : Success
CloseConnection();
[Connection == null]
Success = false
BOOL : Success
SetLoginState(success)
REF: Get Calender
UpdateGui()
Controller
Domain
DAO
DATA
www.websequencediagrams.com

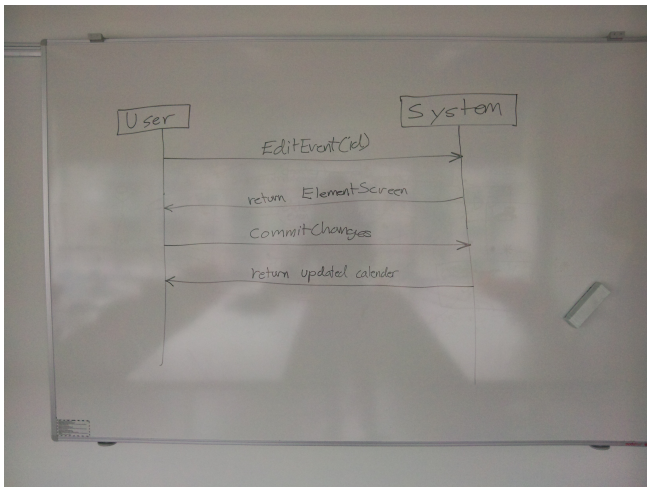

Create Event
Controller
Domain
DAO
DATA
CreateEvent(name, time, desc)
CreateEvent(name, time, desc)
Connection = OpenConnection();
alt
[Connection != null]
Query(name, time, desc)
BOOL : Success
CloseConnection();
[Connection == null]
Success = false
BOOL : Success
CreateTask(name, time, desc)
UpdateCalender()
BOOL : Success
opt
[Success == false]
LogException()
UpdateGui()
Controller
Domain
DAO
DATA
www.websequencediagrams.com



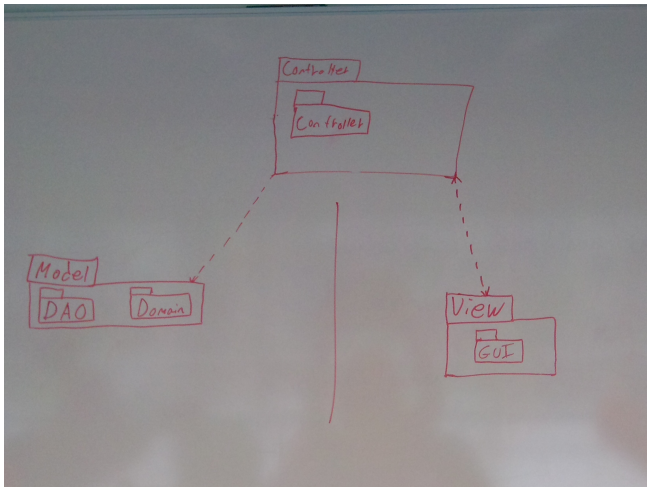
Communication diagrams



Simple sketchy sequence diagrams



Package Diagram



The five GRASP principles

- Creator: The class that either, contains information, records or uses the object to be created should create it.
- High Cohesion: The responsibilities of classes are clear and logical, with reuse, low coupling and maintainainability in mind.
- Indirection: Assigning a class to communicate through(like controller) to lower coupling and increase reusability.
- Information Expert: Assigning responsibilities where the related information are.

- Low Coupling: Make as little dependency as possible between classes to minimize the impact of change and increase usability.

Analysis of GRASP principles in our design
To assses whether our planned design comply with the GRASP principles, the overall design pattern and responsiblity assignments are analysed. As Model-View-Controller is used as the primary design pattern, eventual violations of the low coupling and high cohesion principles are done in the individual components. The View part is poorly planned at this point, and does only involve displaying and parsing of data. However, when it is planned in more detail, focus shall be on using states and polymorphism to implement the variations in methods. The controller could have a higher degree of cohesion, judging from the name of "DrawCalender()". The name should be changed to "DisplayCalender()" or "ViewCalender()" to indicate that the responsibility of the controller is not drawing the calender. The controller handles all system events. The Model has information about the domain and generally acts as both the information expert and the creator. However, the model only contains the calenders, not the events. Therefore it is an option to make calender create the event, because it contains them. All the information comes from the gui, making no component information expert in this case.
Changes proposed:

- Changing method DrawCalender() of Controller to ViewCalendar().
- Assigning the creation of event to Calendar instead of Model, because it contains the events.
- When designing the gui in detail, use polymorphism to implement states.

10 SCRUM

Definition of done: A task is done when the code is working, it have been tested with automated unit tests and have been documented.
Product Backlog

Starting Product Backlog					Remaining as of sprint					
Item	Deatils	Priority	Estimate of Value	Initial estimate of Effort	1	2	3	4	...	6
Set up initial database		1	5	7	0					
As a user i would like to login		1	5	13	0					
As a user i would like to manage events		2	4	19	19					
As a user i would like to share a calendar		3	2	7	7					
As a user i would like to respond to a shared calendar invite		4	2	3	3					
As a user i would like to manage calendars		5	1	7	7					

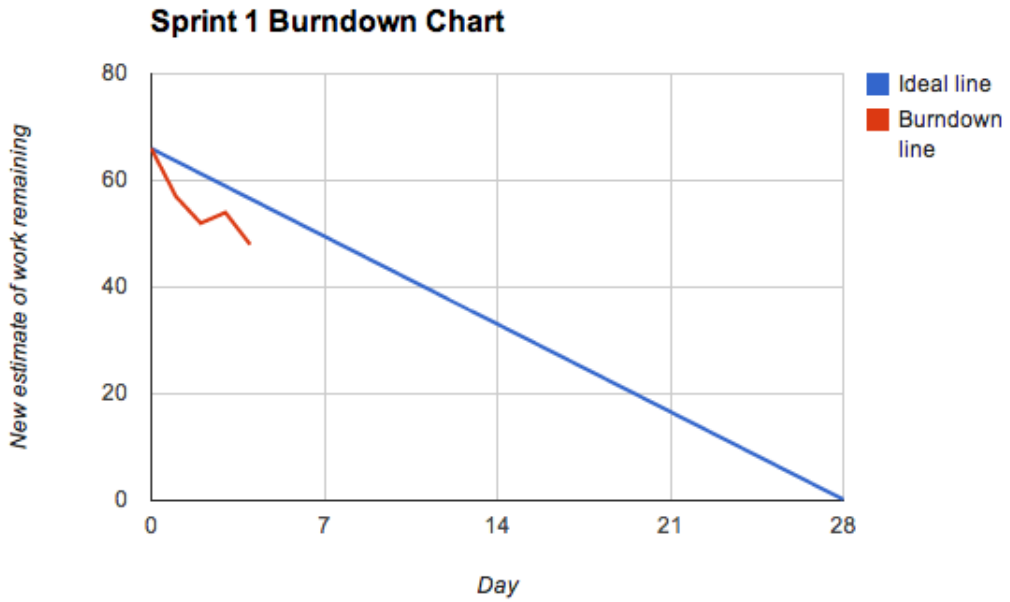
Sprint capacity

Worker	Available days during sprint	Available hours pr. day	Total available hours
Mikkel	20	6	120
Mark	14	6	84
Morten	20	6	120
Dennis	16	6	96
Lea	20	6	120
Kim	20	6	120
		Total:	660

Sprint 1 Backlog

Sprint 1 Backlog				Remaining as of day					
Product Backlog Item	Sprint Task	Volunteer	Initial estimate of Effort	1	2	3	4	...	28
Set up initial database	Figure out which kind of database should be used	Morten, Mark and Dennis	4	0	0	0	0		0
	Create standard tables	Morten and Dennis	14	14	13	12	11		0
	Optimize tables using indexes	Mark	8	8	9	8	7		0
	Test and document	Kim and Morten	10	9	9	9	7		0
As a user i would like to login	Create UI	Lea	6	5	4	7	6		0
	Write client code	Mikkel	6	4	4	5	4		0
	Write server code with authentication	Mikkel	8	7	4	4	4		0
	Test and document	Kim and Mikkel	10	10	9	9	9		0

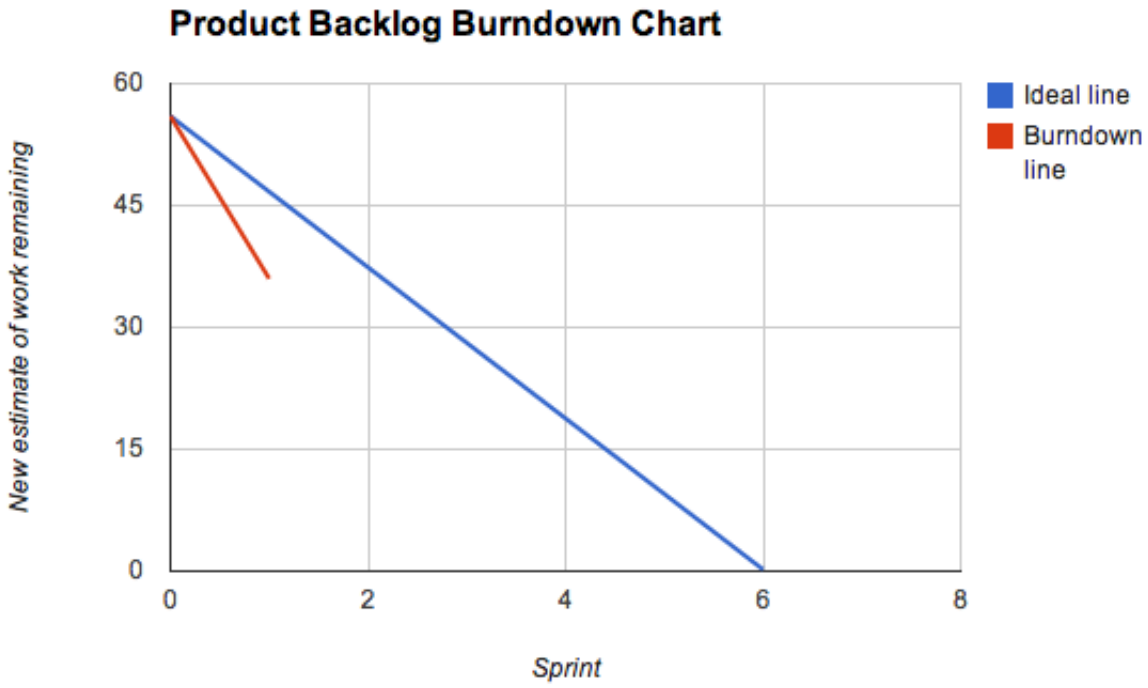
Sprint 1 Burndown Chart



Sprint 1 Review
A walkthrough of the user interface and the login process was showed. The product owner have a few tweaks to the program and the user interface

Sprint 1 Retrospective
The team feels that they could have completed more work in this sprint, if more tasks had been planned.

Product Burndown Chart


Product Backlog Burndown Chart
60
45
30
15
0
0
2
4
6
8
New estimate of work remaining
Sprint
Ideal line
Burndown line