Essay assignment IAK040
2006-05-21

How to make a System Development Project;

”The Archive”
Remote Desktop and File Storage - Server Cluster

- “Server power, in the palm of your hand...”

Student: Peter Carlsson

Index:

Method and Purpose
As a System developer one is taught by many different methods of analyzing the problem we are faced to deal with. We deal with problems and design solutions. The thing we read and study are mostly unfinished models and methods just put in to practice in hope of giving designers good enough statistics of working projects. This area isn't finished by far until all designers agree on using the same choice of methods and models, but we are getting there. The purpose of this essay is to show that some of these great designers are speaking of the same things, and are nearly getting to the point where all of them agree with the method of designing systems. These Waterfall vs Larmans method will give us a insight of what to use.

Introduction
The most difficult state of a system project development must be (what larman describes) as the inception state. In order to do a proper inception state we need to speak clearly to our costumer or client that have come to us for guidance. I will put allot of focus in the beginning of the System project in order to easily describe what will follow and it will probably show more clearly what is necessary to make a working System Development Project on the later iterations.

In this particular essay I'll introduce my current project in work called, ”The Archive” (as it's working name in progress). I'll be pending the typical waterfall models by Checkland and Mathiassen vs Larmans methods to show that either version will help the common System developer Keep in mind that the design of this project document is in that order that Larman uses primarily and that the other authors/designers are just mentioned with what they would have described certain parts of this project. These particular authors/designers are to be marked with a ”red” color in order to sort out which is which. But in a whole this document about ”The Archive” follows our latest project assignment of the course.

You know you didn't understand inception when... (Larman)

* *It is more than ”a few” weeks long for most projects.*

* *There is an attempt to define most of the requirements.*

* *Estimates or plans are expected to be reliable.*

* *You define the architecture (this should be done iteratively in elaboration).*

* *You believe that the proper sequence of work should be: 1.) define the requirements 2.) design the architecture 3.) implement.*

* *There is no Business Case or Vision artifact.*

* *All the use cases were written in detail.*

* *None of the use cases were written in detail; rather, 10-20% should be written in detail to obtain some realistic insight into the scope of the problem.*

Student: Peter Carlsson

- Here starts inception -

”THE ARCHIVE”

VISION (Larman) (Mathiassen calls this the Assignment/purpose)
The Archive is a Desktop On Demand System, which means that users for any reason may need a mobile or outsourced system to access on a clustered central server. There's nothing new about this idea except for the custom made desktop environments and choice of operating systems. User of any kind of profession can connect to this virtual environment and work with almost anything, using the servers powers. The client machines using this system can be anything, from mobile devices or to just a old beaten up computer with a modem, it doesn't matter because the power lies within the servers hardware. The only need is a good stable Internet connection.

By writing a bigger picture or ”vision” of our project we get our assignment clear and show our costumer/client what we are hoping to achieve.

SUMMARY OF SYSTEM FEATURES (Larman (Mathiassen calls this Usabilities)
* Broadcasts Remote Desktops
* Information handling
* On line solution
* Security

ABOUT THE DEVELOPMENT PROJECT OF THE ARCHIVE (Larman)
Administrate The Archive use cases will not be treated in the inception phase since they are the most architectural significant and high-value functions of the project.

Time is not an factor for The Archive, therefor it will not be focus on sequence diagrams and similar. Concerned authorities are notified what information that is stored and what access that is applied to it.

Now it's time to choose which ”rot-definition”, it will give us the most information about this System Project and sort out where we have our ”problem area”. Now what is a Rot-Definition? The decision of changing a information environment and make it a better understanding of the whole. It can for instance be a better way of communication, information processing of infrastructure, which will all make a changes in the information system at the end. It's important that the negotiated and the shared view of what has to be improved, can be as forward and understood as possible.

In order to establish such a rot-definition we need some criteria/methods, like VATOFA, CATWOE and FURPS+...

FURPS+ (Larman)
(Prerequirement)
An Administrator which works with ”The Archive” is responsible to administrate ”The Archive”.

Functional
on line interface
provides remote desktops

Student: Peter Carlsson

saves and stores files
exhaustive log of all actions
exhaustive log of all errors
on line help
encrypted communication

Usability
the interface works and looks like your typical OS environment
personal preferences will be available

Reliability
up time is very high, 365 days a year
time needed to fully recover is one day
backups of the entire system is created every midnight and a clone server cluster is present on another location if the worst scenario possible would happen

Performance
start-up time is low
shutdown time is low
response time is low

Supportability
multi-lingual interface
physical maintenance handled by the Administrator
Design requirement
client/server solution

+ (if needed)

Implementation
Windows Server 2003 or higher
OSX Tiger Server 10.4 or higher

Interface
the system can be used by users on line

Important objects
user
administrator

VATOFA (Mathiassen)

**** Villkår = Demands: The system should always have constant uptime and daily** maintenence.*

**** Användningsområde = Usabilities:** Remote users accessing their desktop environments.*

**** Teknologi = Technology:** A computerized client with network capabilities.*

**** Objekt = Objects:** Users, Software.*

**** Funktionalitet = Functionality:** Provide a interactive workspace on the net.*

**** Ansvar = Responsibility: Administrator with proper computerized communication***

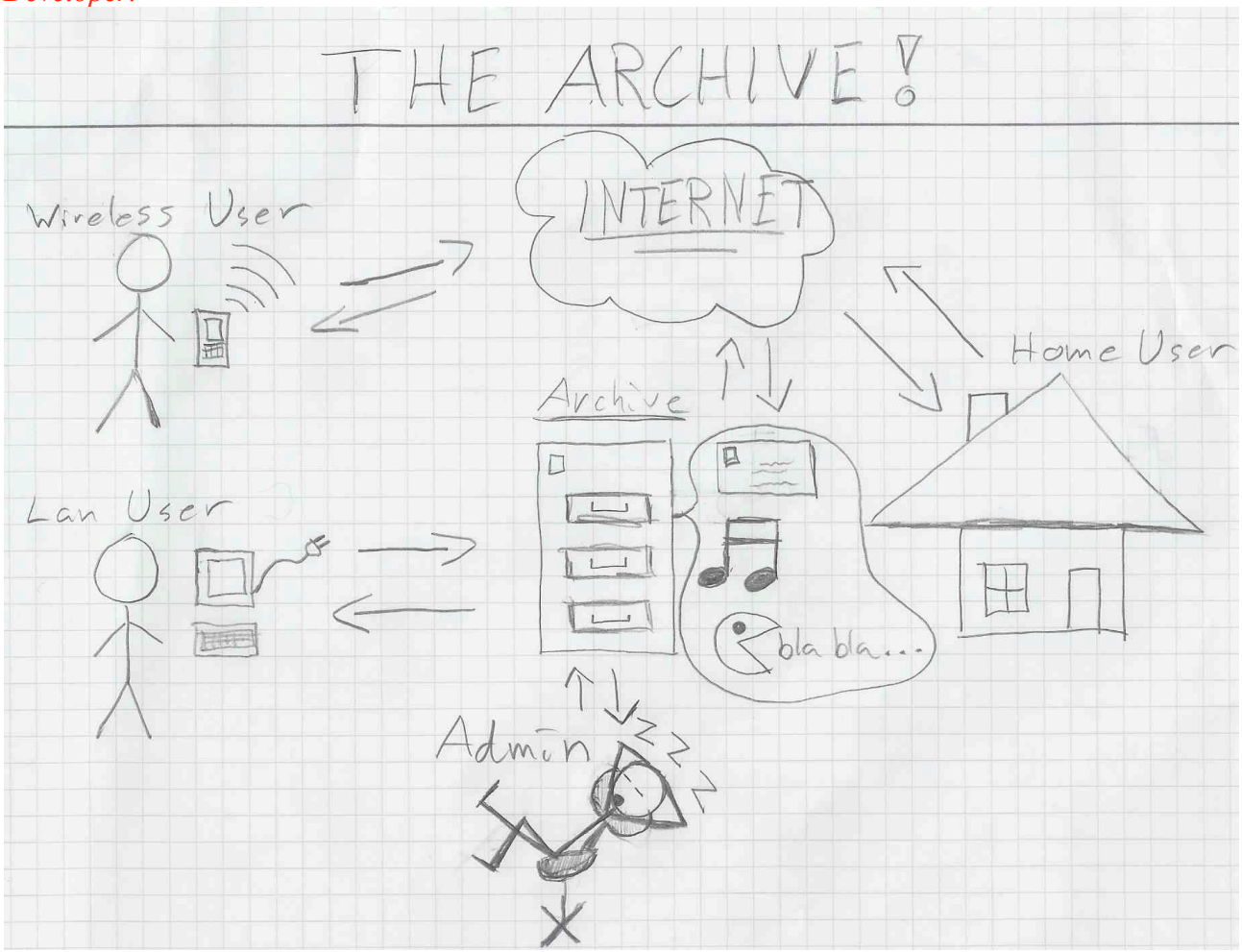
equipement.

***CATWOE** (Checkland)*

**** C = Client: A student, a business worker and so on...***
**** A = Actor: User or Administrator.***
**** T = Transformation:***
**** W = Weltanschaung (World picture): Showing our system with the rest of the world***
**** O = Owner: The company***
**** E = Environment: Interactive Desktop Environment***

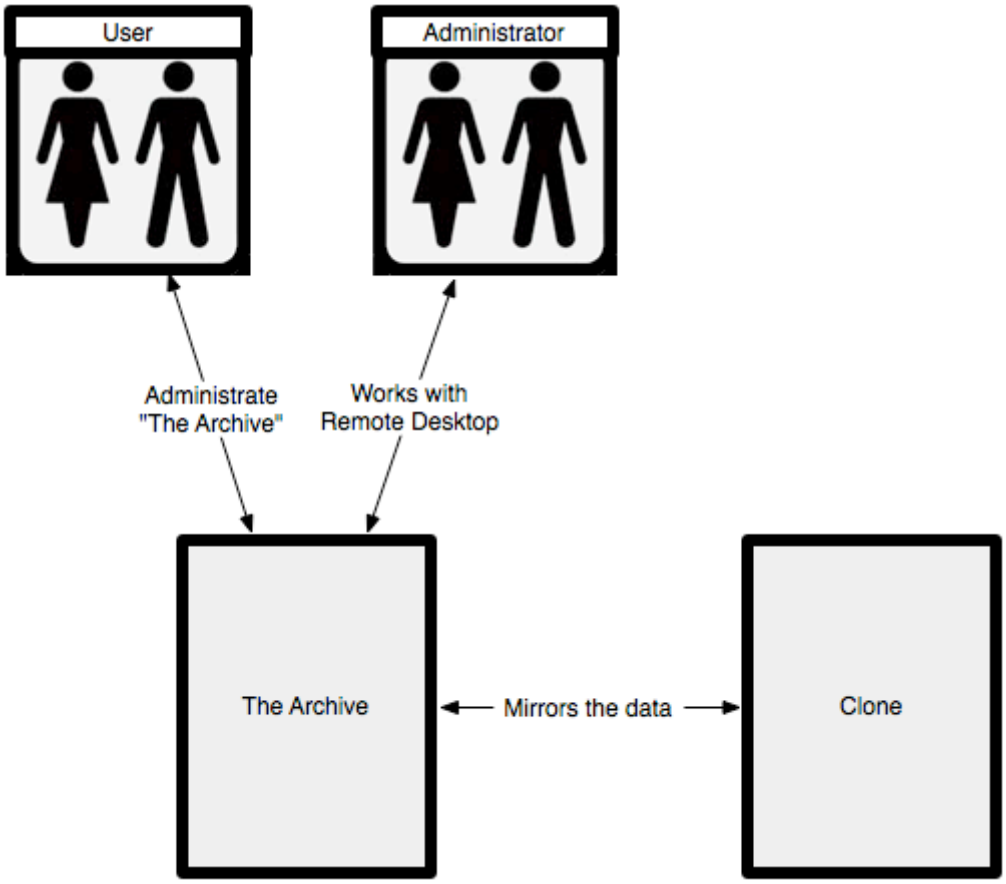
RICHPICTURE *(Checkland and Mathiassen)*

Is another method to describe a ”problem area”. The System Developers purpose of this is to have a understanding communication towards his client in search of this design help we are providing. By doing a rich picture, we can figure out what the problems are and how to view them from both perspectives. When it's ”finished” the second step is to the client/costumer him/her self to show this to his colleagues to meet them with a language they all understand in contrary from the System Developer.



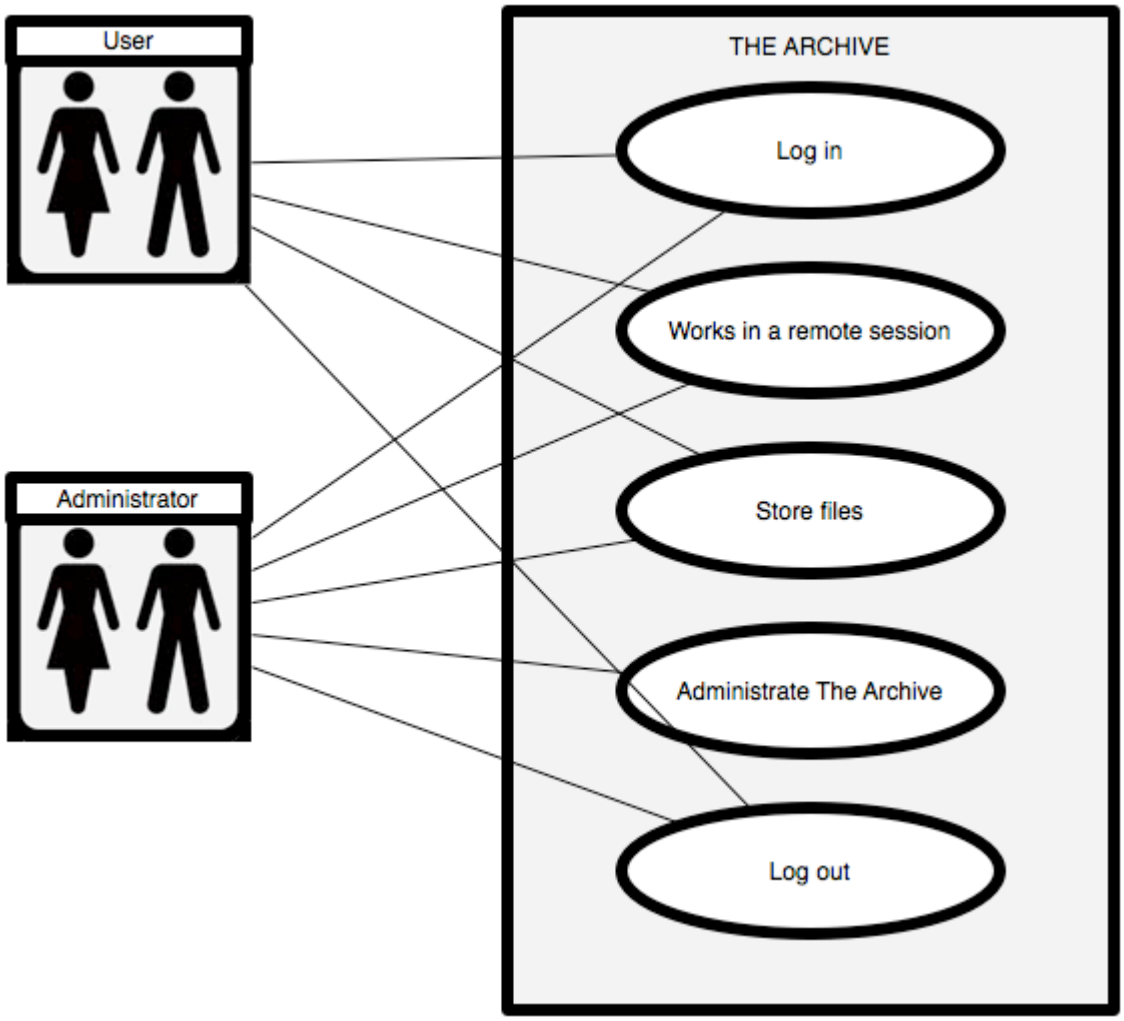
It's recommended that both designer and client/costumer works with this document, preferably on a piece of paper using just a pen and maybe some other drawing tools. There are no rules except keeping it as simple and summarized as possible. The technical aspects isn't that much of importance for the client/costumer just the problem areas.

CONTEXT DIAGRAMS
INFORMATION FLOW DIAGRAM (Brown)



Used in this project but isn't actually a Larman method, he usually use huge domain models describing everything but in this project we use a Information Flow Diagrams to show how the information goes around in our system. You can see this model mostly in Brown.

USE CASE DIAGRAM (Larman, Brown)



Use cases are defined to satisfy the goals of the primary actors. Choose the system boundary, Identify the primary actors, Identify the goals for each primary actor and define use cases that satisfy user goals.

SCENARIOS (Larman)

User types account name and password in client software
User logs in with client software
User access his/hers customized desktop environment
User stores files
User delete files
User updates files
User logs out with client software
Administrator types account name and password in client software
Administrator logs in with client software
Administrator access his/hers customized desktop environment
Administrator adds a new User
Administrator removes a User
Administrator stores files
Administrator delete files
Administrator updates files
Administrator configures and checks systems on ”The Archive” with administrative tools
Administrator logs out with client software

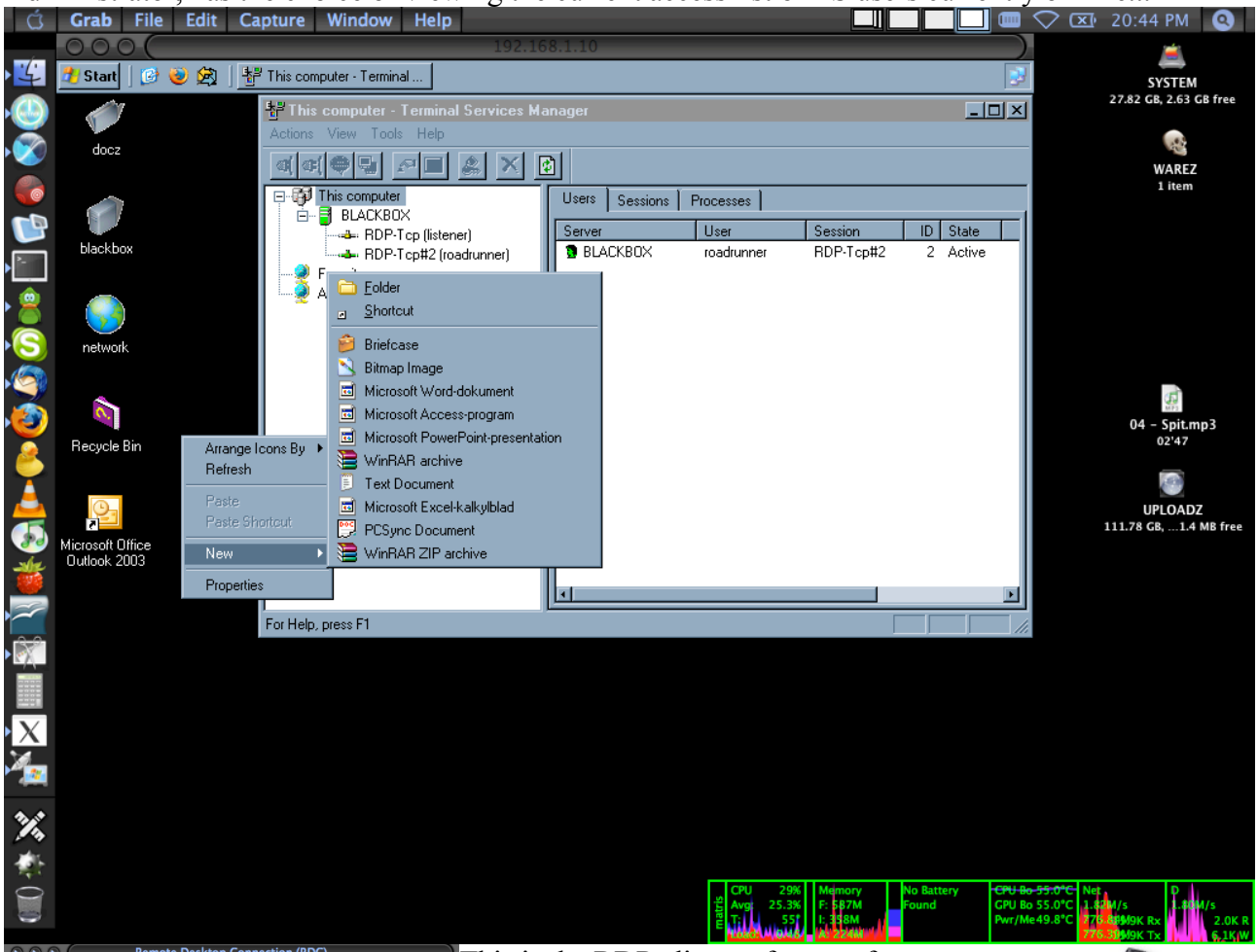
With these we can make for example state chart diagrams later on...

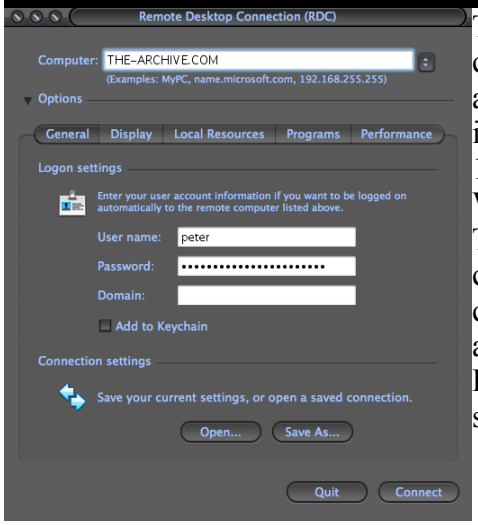
Now Larman tells us to begin structure of a code and start to design programs right away in the inception state, by this method we can start out small and keep updating them until we are satisfied in one of the iteration states. Checkland or Matthiasen preach that the waterfall method of designing models and states of every functions must be finished before even touching a code or whatnot, Larman counter this with statistics showing that the waterfall method isn't sufficient enough and have very high failure I say both methods work depending on what kind of system we are developing.

In this inception state we don't use code because the programs are finished and already constructed, we are just puzzling this together. Making this just as sufficient with the waterfall method or without, either way we get all the information we need to start designing this.

INTERFACE (Larman, Mathiassen)

This is a simple emulation of a user called ”roadrunner” accessing a server called ”blackbox”, it's what one usually see on a computer screen using RDP's. The operating system is Windows 2003 that's being remote accessed to. This works the other way around as well as using this alternate desktop environment in full screen and with more options like local map sharing and local printer use etc. Choices are endless. This particular session shows how the Administrator would view from his/her remote access to the server, in this case we can clearly see that user ”roadrunner” which is the Administrator, has the choice of viewing the current access list of TS users currently on line...





This is the RDP client software of choice, which only use the type TS as it's protocol. This particular setup is a Macintosh ibook using OSX 10.4 with a remote session against a Windows 2003 Enterprise server. The remote desktop session uses a customized desktop interface for this current user called roadrunner. It's adapted to be used with a small PDA mobile called the qtek1010 and can also be used like this shown on this picture above. The choices are endless...

DICTIONARY (Larman)

Client	– A computerized device able to communicate with other computers
Server	– Dedicated computer that provides services for clients
Cluster	– layers of hardware or computers in this situation
The Archive	– Symbolic name of the server cluster
User	– Person in front of his/hers client
TS	– Terminal Services (frequently used remote type of application for office environments)
RDP	– Remote Desktop Protocol
Citrix	– Overlay for TS or other RDP protocols (provides encryption and more stability etc)
PC	– Personal Computer (mostly PC-architectural machines from the 80:th's heavily modified with todays standards more refered as x86)
Mac	– Macintosh (not referred as a PC or x86, more advanced type of machinery and new standards of computer hardware)
PDA	– Short for personal digital assistant, a hand held device that combines computing, telephone/fax, Internet and networking features.

ENCLOSURE (Larman)
FIND PRIMARY ACTORS AND GOALS

Who starts and stops the system?
Administrator
Who does the system administration?
Administrator
Who does user and security management?
Administrator
Is ”time” an actor because the system does something in response to a time event?
No. we don't create any sequence diagram
Is there a monitoring process that restarts the system if it fails?
Yes, handled by Administrator
How are software updates handled?
Administrator
Who evaluates system activity or performance?
Administrator
Who evaluates log?
Administrator
Who gets notified when there are errors or failiures?
Administrator

- This takes care of the inception state of this project -

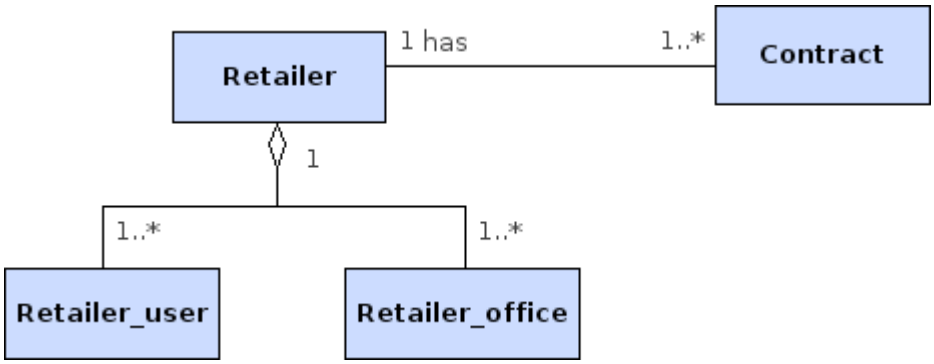
Now without getting to concrete or caught in the oblivion of models and heavy words, we can move on to the iterations. The inception state gave us just enough information to start designing this system and put a ”should be done” stamp on every goal within the project.

In order to start making heavy designs of the System we are making, a Domain model or Class diagram is drawn. This gives us a broad view of which classes we are working with and which should be realized.

Because of this project already being finished by software and planning from other companies making this solution, we'll take another project to see some difference and look at those models instead. This will give us a better view of the ”coding” face that usually comes up in other projects. Here is one example from another system called WRIS:

- Here starts the iterations -

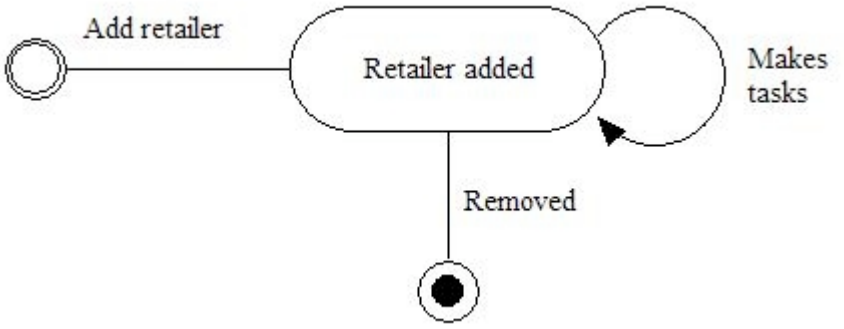
STATIC DOMAIN MODEL / CLASSDIAGRAM (Larman, Brown, Mathiassen)



Here is a Domain Model/Class diagram of a another project called WRIS. This shows us what that particular system does. Here we can clearly see that it handles Retailers. The Retailer_user describes who the person is, Retailer_office is where the person is or what company is it, they are part of the ”super class” inheriting the Retailers attributes. And on the right we have the Contracts in which has something to do with every Retailer being apart of this system in the first place. We could just a well imagine our ”User” from ”The Archive” being described here working with the remote desktop.

DYNAMIC DOMAIN MODEL (Brown, Mathiassen)

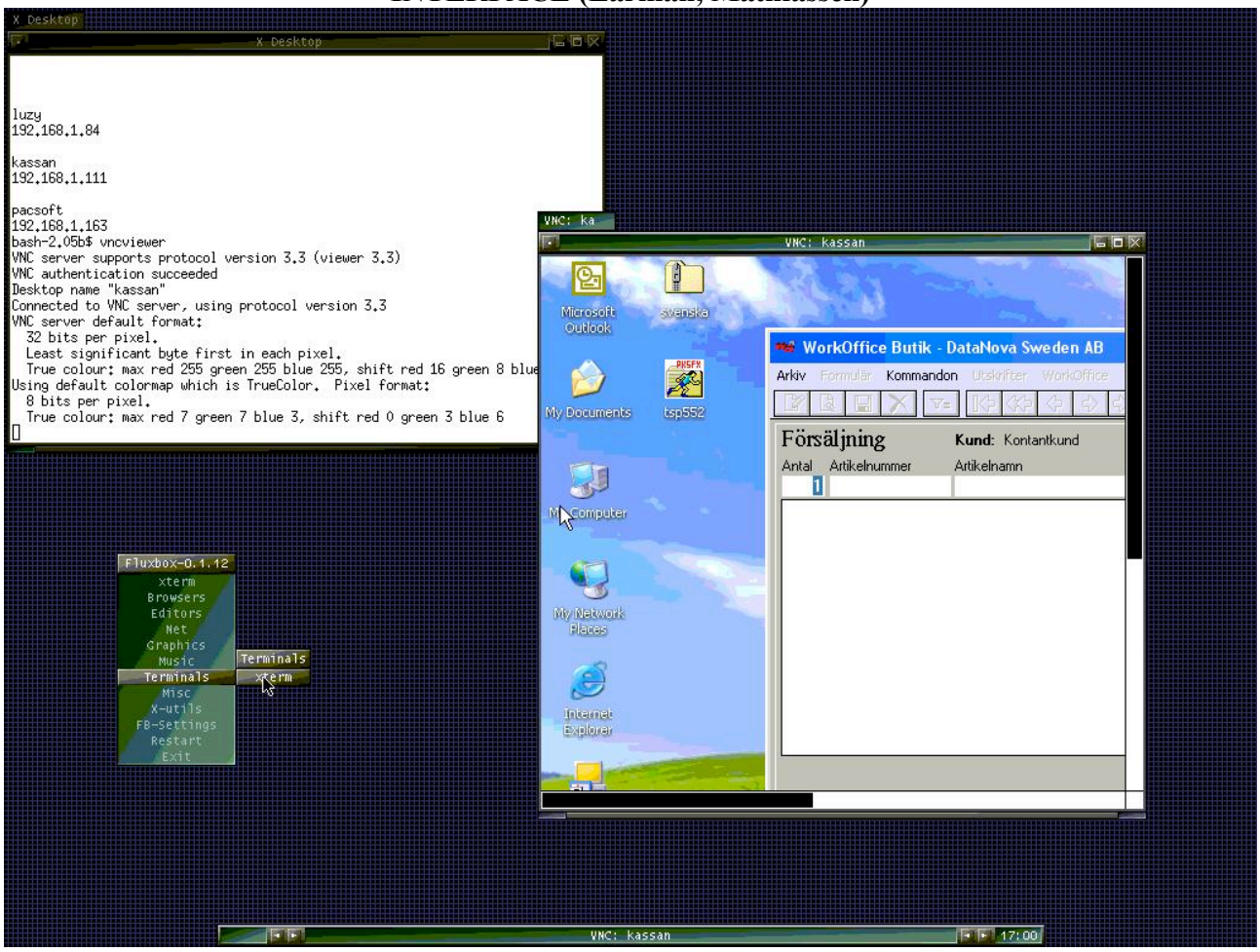
This diagram shows the behavior of the class Retailer in the system.



This would also be called a state chart diagram, describing a sequence from life to death. And here we could imagine our ”Administrator” adding and removing a ”User” in ”The Archive”. (Checkland/Mathiassen)

In the iterative face we once more go through our material looking for changes. In ”The Archive” we now maybe see more features available for our client. New words to the dictionary, more use cases or other models. Even the interface may get some new looks?

INTERFACE (Larman, Mathiassen)



Now we can use a unix client to access our Remote Desktop in a Windows OS environment. Here we see a User working with a company software for sales.

DICTIONARY (Larman)

adding of a new word...
...

Mac – Macintosh (not referred as a PC or x86, more advanced type of machinery and new standards of computer hardware)

PDA – Short for personal digital assistant, a hand held device that combines computing, telephone/fax, Internet and networking features.

VNC - Remote control program, works on almost any OS platform.

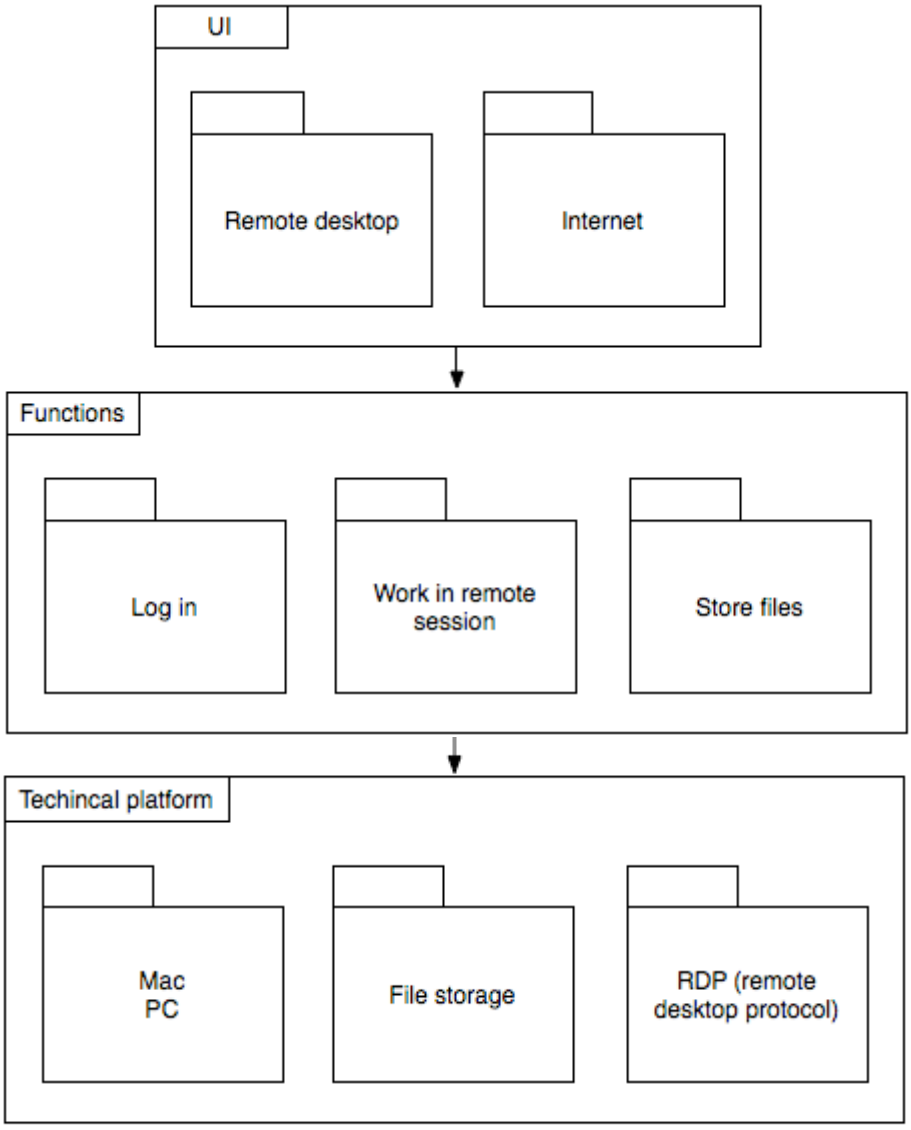
...

In short we need to look back and check our inception material again and again until we are certain that we can't do much better than this...

SYSTEM STRUCTURE – PACKAGE DIAGRAM (Larman, Mathiassen)

The package diagram shows the basic structure and layers of the system including user interface, domain and technical services.

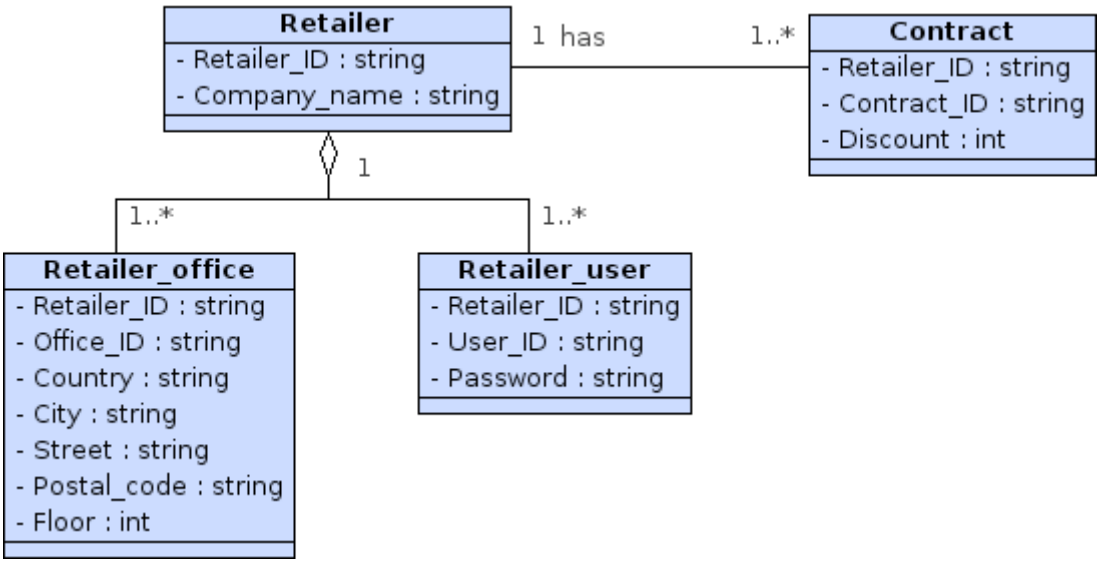
The user interface (UI) is the interface to the user, here it´s represented with a remote desktop application using the Internet The functions layer shows what the application is able to perform. The technical platform layer shows what you need to get the applications running and on what sort of technology.



And now for the designing state, we go back and take a look on that WRIS system we saw earlier, now seeing that we now know what we need to find in order to write a program out of it.

COMPONENT STRUCTURE – DESIGN CLASS DIAGRAM (Larman, Brown)

The design class diagram shows all the attributes that are going to be stored in the database. We will create our database scheme from theses classes and attributes.



*We later use this class diagram to make a program out of it. In the attribution table under each class we now know which variable names to use and which relations they have to each other. We also see how many relations each one of these classes has, 1 or many = *. Which primary key to use in order to get information from it and so on. Later on operations beneath the attributes may come in handy to point out what each class are doing, but it isn't always necessary to write them in these kinds of designs (Brown)*

DATAMODEL (Larman)

Retailer info
WRIS

Retailer_ID	***Office_ID***	***Street***	***Postal_code***	***Country***	***etc***
R1	MC Spec	The Road 2	324 54	US	...
...	...	...	...	...	...
R5	Super MC	The Avenue 12	433 94	UK	...

WODO subsystem

SystemID	***SystemName***
S1	Economy
...	...
S4	Sale

This is a part of the newly created Data model for the WRIS database in the iterations it keeps on growing depending on the code being designed or if new problems/solutions comes to mind.

- This takes care of the iterations of this project, from here it keeps growing -

Now we have everything we need to make this project grow until we are satisfied/deadline approaches or the costumer/client begs for more functions or so.

Summary

All these methods and analyzing strategies are necessary to make a good enough design. As a designer of systems such as this we are depending on correct information being brought forward by brainstorming and modeling. Without this we would constantly be in a trial and error face without knowing how to get around problem areas and solving them once and for all. As mentioned before, nothing is really finished but it's a plan in making something work, as these authors and the their teachers have been analyzing this for years on how to make the best design. Still it's a scratch on the surface and we are getting more efficient of designing systems. It now seams that Larman is the person knowing what to do accordingly to his research and statistics over huge system developments, and he makes good points of why this is. Though models and constant analyzing will give allot of data, one should always remember that the practice of the system should be taken in to mind so not just planing will get to be prio number one.

Sources

*** Craig Larman: "APPLYING UML AND PATTERNS, An introduction to Object Oriented Analysis and Design and Iterative Development, Third Edition."**

*** Lars Mathiassen, Andreas Munk.Madsen, Peter Acel Nielsen, Jan Stage: "Objektorienterad analys och design."**

*** David William Brown: "Second Edition, An Introduction to OBJECT-ORIENTED ANALYSIS, Objects and UML in Plain English."**

*** Kjell Engberg: Course material and teachings around the System Development Project.**