Bachelor's Thesis

Degree Programme in Business Information Technology

e-Business Systems

2011

Jarkko Hurme

THE BENEFITS OF USING UML-MODELING TOOLS IN EVALUATION AND TESTING OF ETM SOFTWARE

TURUN AMMATTIKORKEAKOULU

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology | e-Business Systems

November 2011| 68 + 2 appendices

Oskari Kiviniemi , Minna – Kristiina Paakki

Jarkko Hurme

THE BENEFITS OF USING UML- TOOLS IN EVALUATION AND TESTING OF ETM SOFTWARE

This thesis deals with UML - diagrams as part of a database-based software evaluation and testing. Guidelines for this study are based on Electronic Town Hall Meeting –software (eTM) which is part of the EU level PARTERRE: Electronic Participation Tools for Spatial Planning and Territorial Development project.

This report focuses on the UML –diagrams' role in testing and evaluation of Electronic Town Hall Meeting –software (eTM). It first covers the Electronic Town Meeting – software so that the reader has a clear understanding of the software type and its purpose.

The third part deals with the various UML - modeling tools and their usefulness in database-based software testing. The purpose is to bring out the benefits of UML - diagrams in testing and to advise the reader in their use in order to give a clear overview of the UML- diagram types and how to use them in testing and evaluation process of software.

The fourth section goes deeper into the diagrams focusing on Activity diagram and State machine diagram and demonstrates step – by –step on how to benefit from these diagrams in software evaluation and testing an how to utilize this knowledge in one's work.

The final chapter deals with the writer's own experience on the basis of UML - diagrams that were created for the Electronic Town Hall Meeting –software (eTM). Different sections of them are analyzed so that the reader can understand their purpose and realize how these diagrams have supported the testing and evaluation process as well as the test case design process.

In the summary I will talk about the future implementations of this testing method as well as evaluate my own work.

KEYWORDS:

PARTERRE, eTM, UML – Diagram, Testing, Testing methods

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ
TURUN AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma | Sähköisen liiketoiminnan järjestelmät

Marraskuu 2011 | | 68 + 2 liitettä

Oskari Kiviniemi , Minna – Kristiina Paakki

Jarkko Hurme

UML-TYÖKALUJEN HYÖDYNTÄMINEN ETM OHJELMAN ARVIOINNISSA JA TESTAUKSESSA

Tämä opinnäytetyö käsittelee UML – kaavioiden käyttöä osana tietokantapohjaisen ohjelmiston arviointia ja testausta. Opinnäytetyön pohjana käytettiin Electronic Town Hall Meeting – ohjelmistoa (eTM), joka on osa EU tason PARTERRE: Electronic Participation Tools for Spatial Planning and Territorial Development projektia.

Tässä työssä tarkastellaan UML-kaavioiden roolia eTM – ohjelmiston arvioinnissa ja testauksessa. Työssä käsitellään ensin Electronic Town Hall Meeting - ohjelmistoja niin, että lukijalla tulee selkeä käsitys kyseisistä ohjelmista ja niiden käyttötarkoituksesta.

Kolmas osa käsittelee erilaisia UML – mallinnuksen työvälineitä sekä niiden hyödyntämistä tietokantapohjaisten sovellusten testauksessa. Kyseisen kappaleen tarkoituksena on tuoda esille UML-kaavioiden edut testauksessa sekä neuvoa lukijaa niiden käyttämisessä niin, että lukijalle tulee kokonaiskuva UML- kaavioiden hyödyntämisestä osana testausprosessia.

Neljännessä osassa mennään vielä syvemmälle ja keskitytään vain aktiviteettikaavioon sekä tilakaavioon ja edetään askel askeleelta opastaen miten hyödyntää näitä kaavioita ohjelmiston arvioinnissa ja testauksessa, jotta lukija voi näin hyödyntää tätä tietoa omassa työssään.

Työn viimeisessä osiossa tarkastellaan kirjoittajan oman kokemuksen pohjalta luotuja UML–kaavioita eTM – ohjelmistosta. Kyseiset kaaviot puretaan osioiksi niin, että lukija ymmärtää niiden käyttötarkoituksen, sekä sen miten kyseiset kaaviot ovat toimineet kyseisen ohjelmiston arvioinnissa, testauksen tukena sekä sen suunnittelussa injektoimalla testitapauksia tilakaavioihin.

Yhteenvedossa tulen käsittelemään kyseisen testaustekniikan tulevaisuutta sekä arvioimaan omaa työtäni koskien tätä opinnäytetyötä.

ASIASANAT:

PARTERRE, eTM, UML – kaavio, Testaus, Testaus menetelmä

CONTENT

APPENDICES

PICTURES

FIGURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

UML	The Unified Modeling Language is an open method used to specify, visualize, construct and document the artifacts of an object-oriented software-intensive system under development. The UML represents a collection of "best engineering practices" which have proven successful in modeling large and complex systems.
eTM	Electronic Town Meetings provide communities with a modern option to better reach their audiences. These virtual meetings can reduce the time, effort, and transportation involved in traditional meetings as well as engage a more diverse demographic. (Planning Tools Exchange, 2010.)
EU, European Union	International organization of European countries formed after World War II to reduce trade barriers and increase cooperation among its members
PARTERRE	PARTERRE is an EU funded project that aims to demonstrate and validate Demos and eTM – software for spatial planning and development.
State Diagram	A state diagram is a type of diagram used in computer science and related fields to describe the behavior of systems. (Wikipedia, 2011.)
Activity Diagram	Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. (Wikipedia, 2011.)
eParticipation	ICT-supported participation in processes involved in government or governance.
TTM	Theme Team Member, are the members of the staff in eTM that collect the information sent by participants from the tables.
TTL	Theme Team Leader, are the members of the staff in eTM that collect the information sent by TTM: S

1 INTRODUCTION

We live in an era of advancing information technology. Our lives are intertwined with information technology, be it in our work or in our free time, information technology is all around us. Thus it is no wonder that majority of Governments have noticed the possibilities behind information technology and are starting to utilize these opportunities. One of these initiatives is eParticipation.

eParticipation is about connecting the common people with politics and policy-making and making the decision- making process more understandable with the use of information technologies.

In many European countries the turnout for elections has started to decline. There are numerous reasons for this. Voters may feel that their vote does not "make a difference" or that their opinions or concerns are not being heard. That is the reason why Governments are working with citizens to find and test ways of giving citizens more of a stake in the policy- shaping process for example trough public consultation on new legislation.

Information technology offers solution for this by allowing easier access to information about what decisions affect people's lives and how the policy-shaping process works. Information technology also provides tools for communication and interaction between citizens and governance thus increasing transparency and quality of legislation.

The changes to information technology are not restricted on how we use information technology but they also affect how we develop software. Software development has changed considerably over the past decade. As a result, testing has gained more importance and it can be said that the testing phase is one of the most important phases of software development cost wise as well as for measuring quality of the software.

The aim of this thesis is to show how everyone can use UML- modeling tools as part of the testing process and how this method can be used to test and evaluate Electronic Town Meeting- software. (eTM) Therefore the aim of this thesis will not be in test cases that can break the software. Breaking software is relatively easy but that is not the purpose of a software tester. The purpose here is to test the software and confirm that the core of the software works. Not to interfere with all the errors that are found but to avoid those errors while testing and evaluating the core of the system, making certain that the main purpose of the software works flawlessly. Naturally if there is unlimited time and resources to search for all the errors in the software it could be done, though usually this is not the case.

The frame of reference for this thesis consists of theory about Electronic Town Meeting – software as well as getting familiar with different diagram types that are used in UML– modelling. After addressing the theory portion of these two subjects, the study deals with their mutual relationship of how UML– diagrams can be used as a part of the testing process of Electronic Town Meeting – software (eTM) by injecting test cases into the created diagrams. The purpose of this chapter is to give the readers a clear overview of the UML– diagram's benefits in the testing process and to give the necessary information for novice testers on how to use UML– diagrams as part of their own testing processes.

In the final chapter before conclusion the writer's personal experiences are discussed about Electronic Town Meeting – software (eTM) with activity and state machine diagrams. Examples done for Electronic Town Meeting – software (eTM) are shown and analysed. The test cases created by using UML– diagrams are also described section by section.

It is certain that information technology will continue to advance. This will lead to software becoming more complex and elaborate. As more money is spent on software development also software testing will become even more important aspect of the development. Thus new testing methods will be on demand. The testing method described in this thesis is one of those methods. Testing software and creating test cases by injecting them to state machine – diagrams is not only adaptive but it allows to design the test cases even before the software is fully developed.

2 ELECTRONIC TOWN HALL MEETING

The Town Meeting is a form of structured participation in local government practiced in the US region of New England since colonial times and in some Western states since at least the 19th century. (Parterre Booklet, 2011). In these Town Meetings the entire community was invited by government officials to gather in a public place to discuss or provide feedback on specific issues that had emerged. In its modern version – the Electronic Town Meeting – the use of ICT has enabled numerous citizens to participate either directly or indirectly in issue driven political debates. (Parterre Booklet, 2011.)

An Electronic Town Meeting can be defined in two different ways – it can be defined as a virtual meeting or as a meeting that occurs in person aided by electronic tools. (Planning Tools Exchange, 2010.) Both of them share common features like polling and giving the participant an opportunity to voice their own opinions. Be it by participating on conversations or voting about the discussed topics, though there are differences as well. For virtual meeting these electronic tools serve as a way to engage citizens that cannot participate on local meetings because of distance, geography or simply due timing reasons. (Planning Tools Exchange, 2010.)

Electronic tools in an actual meeting on the other hand serve two different purposes. They give a voice to the participant by polling or by participating in the discussion, though their primary purpose is to increase the efficiency of collecting data about the town meeting. (Planning Tools Exchange, 2010). Thus the participant's opinions can be organized in such a way that the main topics that concern them are immediately visible to the organizers of the town meeting. (Parterre Booklet, 2011).

Naturally the Electronic Town Meeting has its own share of limitations as well although they are outbalanced by the benefits (Planning Tools Exchange, 2010.) the main limitations include. People feel that these tools reduce human communication. The technology itself may limit the types of questions. The system may have glitches that affect the Electronic Town Meeting. It may alienate participants with technology. Depending on amount of participants and the setup cost may be high.

The main benefits include. Reduction in transportation as Town meetings can be attended virtually. They are cost effective due advances in technology. Voting is more accurate and participants can only have one vote. Individual votes are secret. Polling is more efficient as votes are tabulated instantly.

2.1 Purpose of eTM

Electronic Town Meeting could be described as a resource that allows communities and policy designers to engage average citizens in facilitated productive discussions to discover the public's priorities. These virtual meetings can reduce time, effort and transportation involved in traditional town meeting as well as engages broader audience. (Planning Tools Exchange, 2010.)

Policy designers are facing tough decisions that affect the lives of millions. These decisions vary from economy, healthcare to environment thus it is important for those policy makers to engage citizens in important decisions. Electronic Town Meeting does this by giving a voice to the citizen in local, regional or national level.

Example of this is America Speaks' 21st Century Town Meeting which was held after Hurricane Karina hit. (Rebuilding New Orleans) Several attempts to make a recovery plan for New Orleans failed due conflict and distrust in government. That was until America Speaks – organization gathered thousands of average citizens to an electronic town meeting where the citizens of New Orleans could participate in facilitated, respectful dialogue with the policy designers that reflected the diversity of New Orleans. This aided in making a recovery plan for New Orleans. As a result decision makers were able to achieve a level of consensus that lead to the release of several hundred million dollars funding from the Government for rebuilding New Orleans.

Naturally there are differences in Electronic Town Meeting - software’s when compared for example America Speaks’ 21st Century Town Meetings -software focuses more on discussion and deliberation among citizen than speeches, question and answer sections while eTM – software focuses more in question and answer sections. Typical agenda of an eTM is structured according to four steps. (Guida alla discussion 2011, 13.)

1. Initial session of information distribution and analysis allowing the attendees to gain more confidence with the system and get familiarized with the question and polling system.
2. First round of discussions are done in small groups, allowing each participant to voice their opinions.
3. A wrap-up session, during which the results of individual tables are summarized and assembled as a list of questions.
4. Final survey, participants answer the formed questions individually by voting.

2.2 How it works

There are numerous solutions on how to run an electronic town meeting with eTM thus the roles mentioned in this thesis are not all necessary though key roles remain the same. (Parterre-Project, 2011.)

Prior to the meeting a discussion guide is sent to the participants as well as the members of the staff. This discussion guide contains general information about the topics that are present at the electronic town meeting. The purpose of this guide is to familiarize the participants with the subject in case it is a subject which they have no prior experience with. Sending the guide to the staff prior to the meeting serves another reason though, it aids the staff to stimulate discussion and to elaborate the questions during the meeting. (Final Report, 2008, 13.)

The event itself is trusted to the producer, who guides the event and directs the flow of discussion. The producer is also responsible for starting the polling sections and is the one to announce the questions to the participants. He will also introduce the guest presenters. (Discussion Guide 2008, 4.)

Participants are divided into 5-10 member tables either by subject or by ensuring similar group layouts for example 2 female participants and 3 male per table. In each of these tables there is one PC and corresponding number of voting pads compared to participant amount in the table. Normally, one of the participants will act as a secretary. The secretary’s job is to write down the key sentences about the discussion or questions that may have emerged during the discussion and then forward the message to the Theme Team Members by using the computer in the table.

When the polling starts the secretary is tasked of making certain everyone in the table has voted by looking at the on screen message on the computer which displays how many has voted from the table. The voting is done with electronic voting pads where the number responds to the option displayed on the projector screen as well as on the table computers. Each participant has one vote and the last button they press will be registered ad their vote thus they have the option to change their answer until the time runs out by pressing a different number on the vote pad.

Theme Team Members are the members of the staff that collect the information sent by participants from the tables. They combine the messages that they receive from participants and forward those to the Theme Team Leaders while keeping track of already sent messages as well as making sure only viable information is forwarded.

Theme Team Leaders are the ones who form new questions from the messages send by Theme Team Members, though this is not their only task. They are tasked with writing a summary of the topics that combines the information from the forwarded messages. This summary will then be forwarded to Report – user.

Behind the scene the Admin transfers the displays of the staff members as well as the participants to the proper screens when they are needed. Admin is also tasked with creating new user accounts to the eTM – system as well as inserting new questions to the polls that are formed during the meeting by Theme Team Leaders. Prior to the meeting it is also Administrators task to insert all the discussions as well as polls and their options to the system not to mention creating the necessary user accounts for the meeting.

Report - user is tasked with compiling all the summaries send by the Theme Team Leaders as well as summarizing the poll results together into one report which will be displayed and distributed at the end of the meeting to the participants. (Discussion Guide 2008, 4).

Alongside with these key roles there is Video user who is tasked with making sure projector works and that the screen displayed is right. Other roles include photographer who is tasked with taking pictures of the participants and the event which can then be added to the final report.

Though like mentioned before there are multiple roles that are not necessary, an eTM town meeting can be done for example with:

- Participants 5-10 / table
- TTM x 1 / table, participant from the table can act as a TTM thus removing need of secretary instead forwarding messages directly to TTL
- TTL x 1 / 2 x TTM, who forms questions and summary which is forwarded to Report – user.
- Report - user can also act as Video user if one is needed, will then need two PC: s though.
- Admin to transfer displays, insert questions, start polling

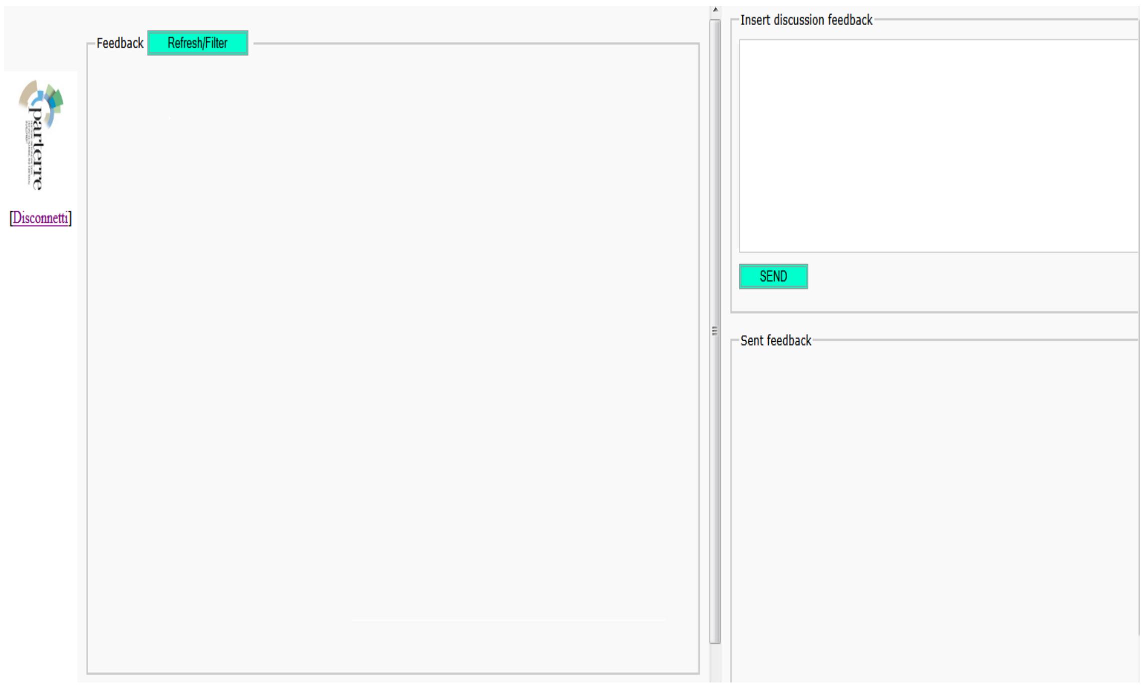
- Producer to lead the event

In the next section I will go over the main roles and describe the key features of those roles as well as their interfaces, thus giving the reader the ability to operate different roles of eTM - software if a need rises.

2.2.1 TTM

Theme Team Members role in a meeting consists of forwarding combined and clarified messages from tables to Theme team Leader while managing in a steady flow of information coming from the tables.

The interface for TTM is highly stripped only having the necessary features to perform the role. On the left side of the screen you have Refresh button. When the button is pressed it will update the text field, allowing the TTM to receive messages from the participants at his own pace to avoid being overwhelmed by the constant message flow. Below the Parterre logo on the left side of the screen is the login out option. On the right side of the screen we have a text field where the TTM can write and copy - paste participant messages after combining and clarifying them. Below that is the send button which will forward the content of the text box to TTL when pressed. On bottom left you see a field where it displays all send messages to the Theme Team Leader, these messages are time stamped which will help to keep track of what has already been processed and forwarded.

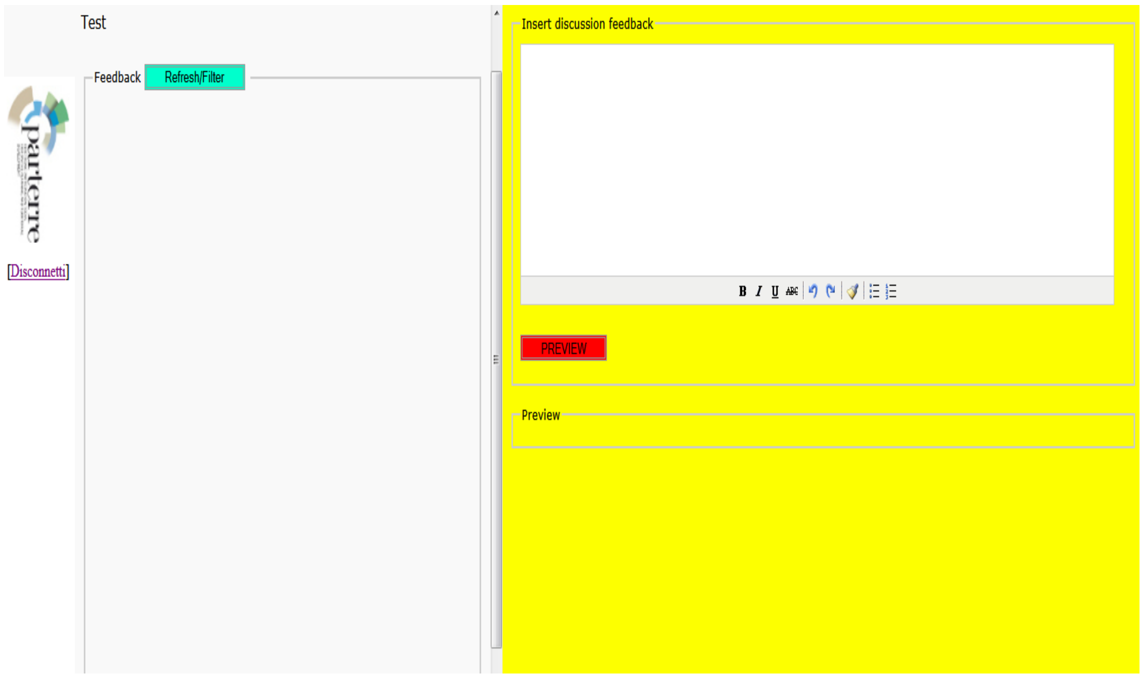


Picture 1. TTM Interface

2.2.2 TTL

Theme Team Leaders role in a meeting consists of forwarding combined and clarified messages from Theme Team Members to Report user, writing synthesis for Report user and picking up new questions from comments to be inserted into the polls while managing in a steady flow of information coming from the Theme Team Members.

Similar to Theme Team Members the Theme Team Leaders interface only has the necessary features, it also shares similar layout with the Theme Team Member interface for example login out is in same place. On the left side of the screen you have the refresh and filter button which will control the flow of messages. Below that is the field where the messages will appear. The messages will be time stamped and they will have the name of the user which send them to you for example clarifying if it’s from TTM1 or TTM2. On the right side you have a text box where you can copy - paste and type the combined information that you have gathered to form summary or new questions to be inserted into the polls. Below that is the preview area which will show what you have forwarded so far.



Picture 2. TTL Interface

2.2.3 Participant

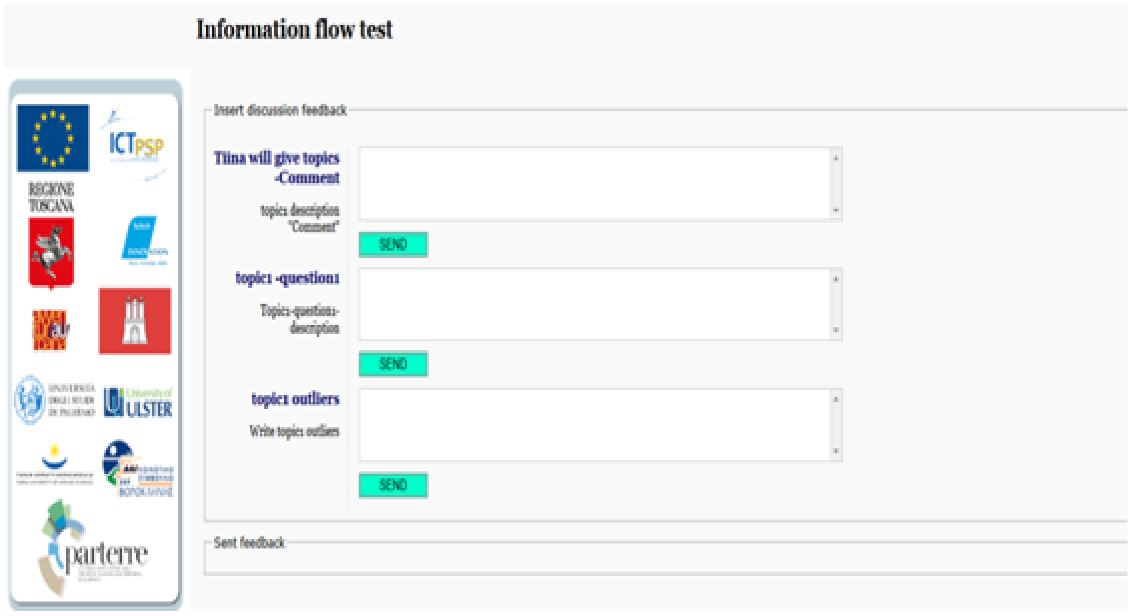
The participant’s role in a meeting consists of forwarding and dividing the comments from the table to different categories. Supervising the voting process in the tables and informing the staff if there are technical issues in the table.

Participants interface looks different to Theme Team Members – interface and Theme Team Leaders – interface. The participant’s area is divided into three different boxes. The top box is for comments about the current topic. The middle one is questions that the participants might have about the subject. The bottom box is for outlier style comments and feedback. After the participant has typed the text he has to press the corresponding send button to send the feedback forward.

Another important interface for participant is the polling. When the polling starts the admin will change the participants screen to display the first question of the poll along with the options. On the top left corner of the screen the participant who is the secretary will see how many has already voted thus he can ensure that all of the participants in the table have voted. Polling itself is simple; every participant will be given a vote pad that has numbers from 0, 1-9. Zero is usually the joker value thus it is used to cover wider range of options for example if you want to know the participants home country:

1. Finland
2. Sweden
3. Norway
4. Denmark
5. England
6. Ireland
7. England
8. Italy
9. Holland
0. Some other country

Once the polling starts the number the participants presses is her/his vote, though if the participant changes their mind they can press another option as long as there is time left in the question thus the new vote will be registered as that participants vote.


Information flow test

Insert discussion feedback

Tiina will give topics -Comment

topic1 description "Comment"

topic1 -question1

Topic1-question1-description

topic1 outliers

Write topic1 outliers

Sent feedback

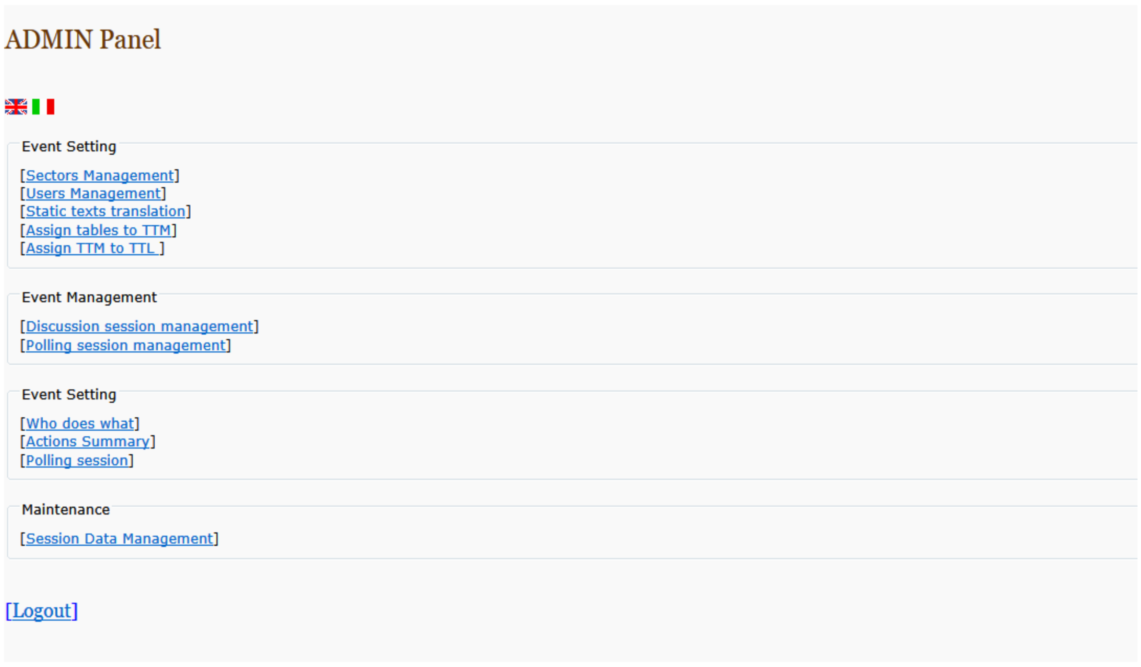
Picture 3. Participant Interface

2.2.4 Admin

Administrator’s role in the electronic town meeting is a vital one. Administrator’s task is to create all the necessary user accounts, sectors, discussions and polls prior to the meeting. When the meeting begins administrator is tasked with transferring people from different screen’s to another, inserting new questions to the polls, starting and closing polls and managing user accounts by adding new ones if necessary.

Administrators interface is divided into four main partitions. The first two partitions could be summed as pre-event management as all the options are aimed at creation. In the first partition you find the options to create sectors or users. You can also assign tables to a Theme Team Member or decide which Theme Team Members are assigned to which Theme Team Leader. In the second partition you can create topics as well as polls.

The third partition contains options for the event. These include action summary from which the administrator can move people to different displays and polling session which contains options to sort polls and compare them as well as option to start the polling and stopping it. The final partition is maintenance related options. From here you can edit the languages as well as delete the database or specific sections for example clear all stored discussions.



ADMIN Panel

Event Setting

[Sectors Management]
[Users Management]
[Static texts translation]
[Assign tables to TTM]
[Assign TTM to TTL]

Event Management

[Discussion session management]
[Polling session management]

Event Setting

[Who does what]
[Actions Summary]
[Polling session]

Maintenance

[Session Data Management]

[Logout]

Picture 4. Administrator Interface

3 UML

As the strategic value of software increases for many companies and businesses, the industry looks for solutions to improve the quality of software while reducing the time and cost to develop them. These techniques include for example frameworks, visual programming and component technology. As a response for these needs Unified Modeling Language was created. (Deer & Moor 2006, 63).

The primary goals in the creation of UML were: (Deer & Moor 2006, 63).

- Provide users with a ready-to-use, eloquent modelling language for developing and exchanging models.
- Provide extensibility and specialization mechanisms to expand the core concept.
- Be independent of programming languages.
- Provide a formal basis to understand the modeling language.
- Encourage the growth of object orientated tools market.
- Support higher- level development concepts such as frameworks, collaborations, patterns and components.
- Integrate best practices

The Unified Modeling language (UML) is a standardized modeling language for specifying, visualizing, constructing and documenting of software systems as well as other non software systems. (IBM Rational UML Resource Center, 2011.)

Modeling Language represents a collection of best engineering practices that have proven Successful in the modeling of large and complex systems. Unified Modeling Language has an important part in the development of object orientated software as well as their development processes. (OMG, 2011.) Unified Modeling Language uses mainly graphical notation to express the design process of the software. That is one of the reasons why UML has become so wide spread. (OMG, 2011).

The tools that Unified Modeling Language offers can aid projects in numerous situation for example they can be used to validate the architectural design of the software, explore potential designs, help project teams to communicate or they can be used to aid in the software testing process.

3.1 History of UML

Identifiable object orientated modeling languages started to appear in the late 1970: s as various methodologists experimented with different solutions to object orientated analysis and design. The amount of these languages steadily rose and during the period between 1989 - 1994 there were already more than 50 languages all with their own notations. This period would later be known as the era of “method wars”. (Sourcemaking).

Three of the most popular methods were OMT (Rumbauch), Booch (Booch), OOSE (Jacobson) each of these methods had its own strengths and weaknesses for example OMT was a simple modeling language that was better at object orientated designs but weak at analysis. (Sourcemaking.)

The development of UML began in the year 1994 when Grady Booch and Jim Rumbauch began their work to merge their methods together due to similar evolution in their methods. (History of UML, 10.)

In the fall of 1995, Ivar Jacobson joined the two in their effort to merge their methods together. This lead to the merging of OOSE with their work, though it soon became clear that it would be too difficult to successfully merge all three methods together. Thus they focused their efforts for unifying their modeling languages for three reasons.   (History of UML, 10).

The methods were already evolving toward each other independently. It made sense to continue that evolution together rather than apart, eliminating potential differences (Xpdian).

By unifying the semantics and notation, they could bring some stability, allowing projects to settle on one mature modelling language.

They expected that their collaboration would yield improvements to their methods, helping them to capture lessons learned and to address problems that none of their methods had previously handled well. Their combined effort resulted in the release of UML 0.90 and 0.91 documents in June and October of 1996 thus the "method war" had finally come to an end. (Xpdian).

3.2 Building blocks of UML

Unified Modeling Language has a vocabulary which consists of building blocks (Booch ect. 1999, 24). These blocks are called things, relationships and diagrams. Things are the most basic building blocks. They can be divided into four categories:

- Structural
- Behavioral
- Grouping
- Annotational

Structural things represent physical and conceptual elements that define the model these include class, interface, use case, node. (Booch ect. 1999, 24). For example use case represents set of actions that are performed by the system for specific goal.

Behavioral things represents the dynamic part of the model these include interaction and state machine. (Booch ect. 1999, 25).

For example interaction is defined as behavioral thing that comprises a set of messages exchanged among set of elements to perform a specific task.

Grouping things are a mechanism to group elements together for example package. Package is used for gathering structural and behavioral things together. (Booch ect. 1999, 28).

Annotational things can be described as mechanism to capture descriptions and comments for example note which can used to describe constraints of elements (Booch ect. 1999, 29).

Relationships are another important building block of Unified Modeling Language. They are used to represent how elements are associated with each other and this association describes the functionality of the application. Like things relationships can be divided into four categories:

- Dependency
- Generalization
- Realization
- Association

Dependency describes the relationship between two elements in which a change in the independent element may also affect the other element. (Booch ect. 1999, 30).

Generalization can be described as relationship between elements which connects a specialized element with a generalized one. (Booch ect. 1999, 30).

Realization can be defined as relationship where two elements are connected. One describes the responsibilities and the other implements them. (Booch ect. 1999, 30.)

Association can defined as a set of structural links among objects in the model. (Booch ect. 1999, 30).

UML- diagram is the final building block in the UML- vocabulary; it is the means by which you view all these building blocks.

3.3 Diagrams of UML

Unified Modeling Language consists of 14 different diagram types; each of these diagrams has a different purpose though they share a common design feature. They all allow developers and customers view a software system from different perspective. (Booch ect. 1999, 85.)

The diagram types can be divided into two categories. Structure diagrams emphasize the things that must be present in the system being modeled Behavior diagrams emphasize what must happen in the system being modeled. Interaction diagrams are a subset to behavior diagram; they emphasize in the flow of control and data among the objects in the system.

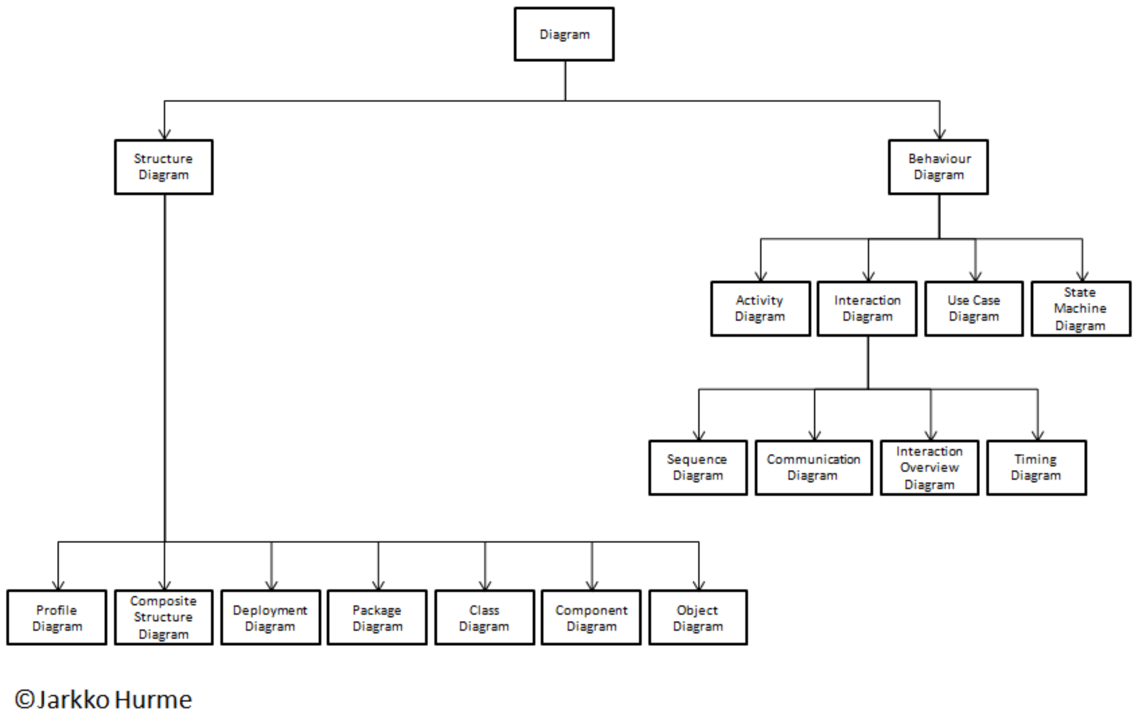
Structure Diagrams:

- Class diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Object diagram
- Package Diagram
- Profile Diagram

Behavior Diagrams:

- Activity Diagram

- UML State Machine Diagram
- Use Case Diagram
- Interaction Diagrams:
    - Communication diagram
    - Interaction overview Diagram
    - Sequence Diagram
    - Timing Diagram



Picture 5. Hierarchy of UML- diagrams

Class diagram shows the classes, interfaces, collaborations and the relationships between them. It is the most common object-oriented approach. Class Diagrams express the static system design perspective (Booch ect. 1999, 31.) Class diagram is typically used in static modeling in one of three ways: (Booch ect. 1999, 98).

- To model the vocabulary of the system

This means deciding which abstractions are part of the system and which fall out of the systems boundaries.

- To model simple collaborations

Collaboration is a set of classes, interfaces and other elements that work together to provide cooperative behavior

- To model logical database schema

You can model schemas for databases that have store persistent information in relational databases thus creating a blueprint for the system.

Class diagrams can also contain active classes which mean they can own processes or manage their own activities. In addition to analysis and planning class diagrams have a central role in system testing as well.

Component diagram shows the components and their dependencies. Component diagrams indicate the static implementation view of the system. They are linked to class diagrams, because the component typically includes one or more classes, interfaces, or cooperation. (Booch ect. 1999, 32.)

Composite structure diagram presents the internal structure of classes or objects and its connections to the other parts of the system. The role of the diagram is to act as a container for the interaction points of various objects of the system and to demonstrate their relationships and configurations.

Deployment diagram shows the run - time structure of the active nodes and their components. Deployment diagram addresses the static deployment view of architecture. They often contain one or more components. They serve as important input for planning and implementation. (Booch ect. 1999, 32.) When modeling static deployment view of a system, you will typically use deployment diagram in one of three ways: (Booch ect. 1999, 340.)

- To model embedded systems

With deployment diagram you can model not only the embedded system but the processors and devices that compromise it.

- To model client/server systems

Deployment diagram can be used to model the topology of client/server systems thus making clear separation between client side and server side interfaces.

- To model fully distributed systems.

Deployment diagram can be used to visualize the current topology and distribution of components to reason about the impact of possible changes to that topology.

Object diagram presents objects and the relationships between them. It presents static snapshots about the objects found in class diagrams. Similar to class diagram these diagrams address the design view of a system but from a real or prototypical case perspective. (Booch ect. 1999, 31.) Object diagrams are also important for constructing the static aspects of systems trough forward and reverse engineering. Object diagram is commonly used to model object structures. (Booch ect. 1999, 169).

Package diagram describes the dependencies between packages that make up the model. They are used in analysis and design for example by illustrating the functionality of software with packages that contain use cases.

Profile Diagrams provide a means of extending the UML by defining custom stereotypes, tagged values and constraints. (SparxSystems, 2010.)

Activity diagram is a special case of the state diagram, which shows the activities between the fluxes. Activity diagram reflects a dynamic view of the system. It is particularly important in modeling the system functions and presenting the flow of control among objects. (Booch ect. 1999, 32.) Activity diagram is also used for system testing

UML – State machine diagram shows the state machine, its different modes, transitions, events and activities. UML – State machine diagram reflects the dynamic view of a system. It is an important tool for modeling the behavior of classes and interfaces. UML – State machine diagram emphasizes event-ordered behavior of an object, which is especially useful when modeling reactive systems as well as testing the software. (Booch ect. 1999, 32.)

Use case diagrams describe a series of use cases, actors and their relationships. Use case diagrams are particularly important in modeling and organizing the systems behaviors. (Booch ect. 1999, 31.) Use case diagram is typically applied in one of two ways.

- To model the context of a system

This involves asserting the whole system to clarify which actors lie outside of the system and interact with it.

- To model the required system.

This involves specifying what the system should do independent on how the system should do it. (At1Ce.)

Sequence-, Communication-, Timing-, Interaction overview- diagram are all interaction diagrams. They present the interactions, objects and their relationships as well as the messages between them. Interaction diagrams indicate a dynamic view of the system

Communication diagram emphasizes on the interactions between objects and on the sequence in which messages are received and send. It is almost identical to sequence diagram though the difference is that in sequence diagram the focus is on the chronological order of the messages.

Sequence diagram shows how processes operate with each other and in what chronological order. (Booch ect. 1999, 210).

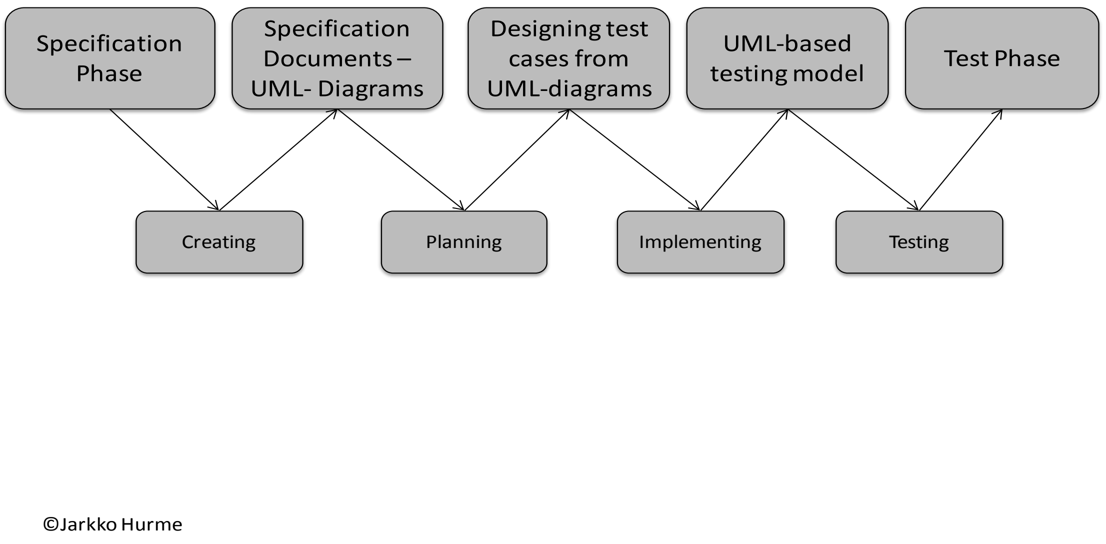
The sequence diagram is beneficial for flushing out the system design as well as testing the software because it shows the interaction between the objects in the system in the time order that the interactions take place.

The interaction overview diagram is similar to the activity diagram both visualizing a sequence of activities. The difference is that the individual activity in the interaction overview diagram is pictured as frames, which can contain interaction - or sequence diagrams. (Wikipedia, 2011).

The timing diagram is used when the purpose is to describe the time-dependent interactions. The diagram shows how an object changes its state at certain points in the timeline. Timing diagram is often used in embedded systems design, but they can also be used for transaction-based systems development.

4 USING UML – TOOLS IN TESTING

UML - based testing is related to functional testing, which is done with limited knowledge about the system or without any prior knowledge about the system. Test cases are designed without the source code and are based on specification-, and requirement documents as well as UML - diagrams.



Picture 6. Simplified structure model of UML- testing

The software tester can create a fictional model by analyzing the specification documents and UML – diagrams, according to which the software can be tested in the future.

The purpose of the UML - diagrams that are drawn in the specification phase is to describe the structure and operation of the software from different angles. The amount of diagrams created equals the benefits gained. Thus the number of diagrams that have been prepared in specification phase, the easier it will be for the tester to review the system from different angles and get a clear overview of the system. Therefore the tester can design more elaborate and comprehensive test cases.

When the system has not yet been built, the tester is able to affect the very design of the system and take into account the anticipated problem areas. Therefore UML- based testing is particularly suited for software projects where the aim is to build new software or to add a new feature to existing software.

4.1 UML – State machine

UML – State machine diagrams can be used in multiple situations of software development for example they can be used:

- To describe the behavior of a system

UML – State machine diagrams can describe all of the possible states of an object during its lifetime as events occur. States are defined as a condition in which an object exists and it changes when some event is triggered. Thus one of the most important purposes of UML – State machine diagrams is to model the life time of an object from creation to termination.

- To model the dynamic nature of a software.

UML – State machine diagrams have a distinguishing characteristic for modeling reactive systems due to state changes being dynamic in nature. Reactive systems can be defined as a system that responds to events which are internal or external factors that influence the system.

- Reverse and forward engineering

UML – State machine diagrams can be used to forward engineer especially if the context of the diagram is a class

- To create test cases by injecting them into the state machine diagram with the aid of equivalence classes.

UML – State machine diagrams can be part of the testing process by creating test cases that are injected into the diagrams. The testing process should start by clarifying the following points:

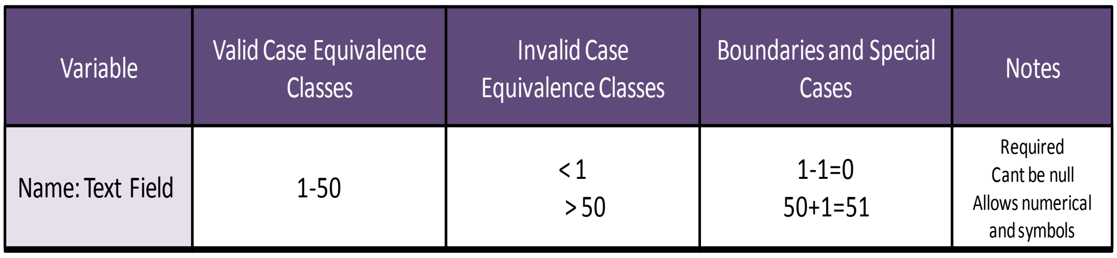
1. Identify the important objects.
2. Identify the states.
3. Identify the events.

Once those main points are clarified, it is a lot simpler to draw the UML – State machine diagram. The diagram can be the whole system or a segment of the system. Large systems are best to divide into smaller segments.

After the system or segment is drawn, a state is selected from it. In this example the Name state is selected.

Selecting a single state serves the purpose of allowing better analysis of that state which will aid in the creation of equivalence classes as well.

Figure 1. Equivalence classes



Variable	Valid Case Equivalence Classes	Invalid Case Equivalence Classes	Boundaries and Special Cases	Notes
Name: Text Field	1-50	< 1 > 50	1-1=0 50+1=51	Required Cant be null Allows numerical and symbols

Equivalence classes are subsets of data processed by the software, which should cause similar software behavior. They allow the software tester to avoid testing redundancies thus saving time as the tester does not need to check each input data but only specified representatives of the class.

The term equivalence class comes from mathematics, where it describes data with same characteristics. For example, if two numbers are congruent, they belong to the same equivalence class.

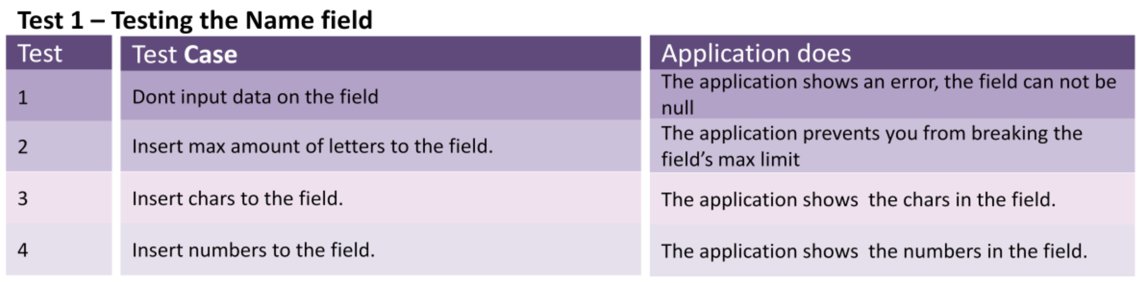
Test data belong to the same equivalence class if the result is expected to be the same, Successful or fail. Boundary values create a separate class as the software is most likely to fail at the transition from one class to another.

There is no good way to find all equivalence classes for specified software but there are some rules, which can help us to do that. At first we divide data into equivalence classes according to the software's documentation. It should give us at least some idea which data will be processed correctly and which data will not.

When considering boundaries we should not only think about obvious borders between equivalence classes but also the data that is not strictly specified in the documentation. This method requires that the tester has some domain knowledge of the subject for example knowledge about ASCII.

With the equivalence classes it is easy to create restrictions to the test cases as well as visualize them better for example look at the equivalence class table on Picture 7. It can be seen that the Name: Text Field cannot be a null value nor have less than 1 char length or more than 50 char length as maximum. The field's type is text field and it can also contain numbers as well as symbols.

With the use of equivalence classes creating test cases for the object; state by state, will become less complicated.


Test 1 – Testing the Name field

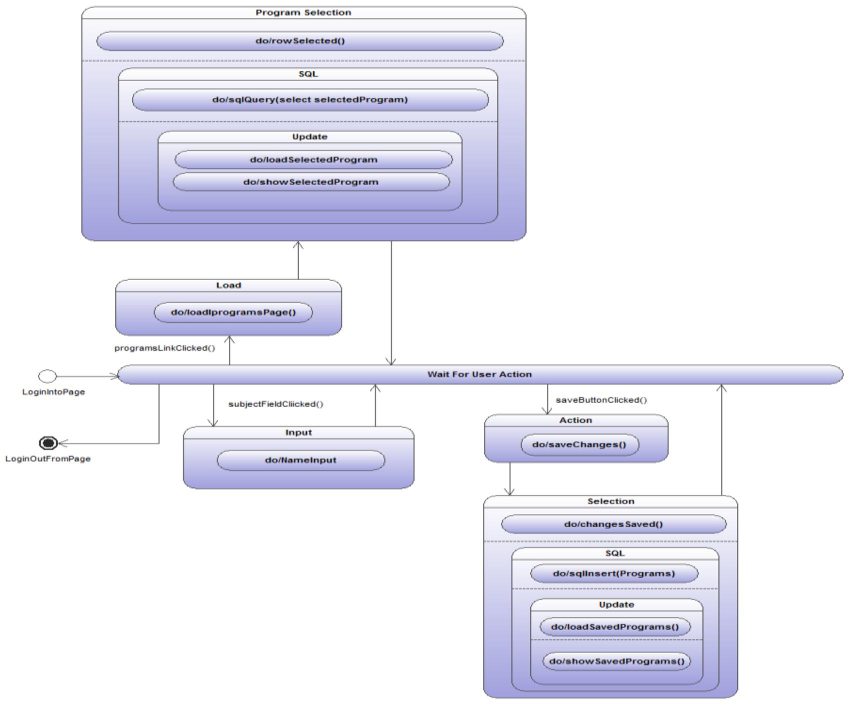
Test	Test **Case**	Application does
1	Dont input data on the field	The application shows an error, the field can not be null
2	Insert max amount of letters to the field.	The application prevents you from breaking the field's max limit
3	Insert chars to the field.	The application shows the chars in the field.
4	Insert numbers to the field.	The application shows the numbers in the field.

Picture 7. Test Cases

State machine diagrams have a limited amount of elements in their notation. The basic elements are rounded boxes that represent the state of the object and arrows indicting the transition to the next state. The activity section of the state symbol describes what activities the object will be doing in that state. Initial state is marked with a circle while final state is marked with an encircled circle.

In these state machines I use SQL – sentences to describe the interface of the states like I have been taught. This will make them more understandable as Unified Modelling Language does not contain a solution to display them.

The following is an example of a diagram where the states of Program object is analyzed. The first state is a wait state from where the process starts. The next states are events like *name field clicked*, *program link clicked*, and *save button clicked*. These events are responsible for state changes in the object. During the life cycle an object goes through those states and there might be some abnormal exists as well. This abnormal exit may occur due to some problem in the system. When the entire life cycle is complete it is considered as complete transaction.



Picture 8. Example of UML – State machine diagram

4.2 Activity Diagram

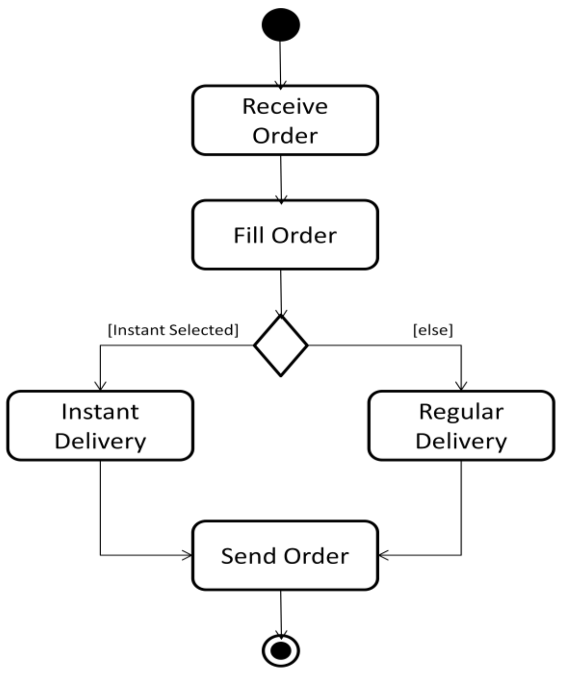
Activity diagrams present the workflow behaviour of a system. They describe the state of activities by displaying the sequence in which these activities were performed. Activity diagrams are not only used for visualizing dynamic nature of a system but they are also used to analyze use cases by describing what actions need to take place and when they should occur.

Other purposes for activity diagrams are constructing executable systems through forward and reverse engineering and aiding in the testing process of software.

The main element of an activity diagram is the activity itself. After identifying the activities on the system and their actions, it is important to learn how they are associated with conditions as well as constraints. Once those elements are analyzed it is best to form a mental layout of the entire flow which then is transformed into an activity diagram.

The notation of activity diagram is similar to UML – state machine diagram. Encircled black circle represents the final state while solid circle is the initial state. Rounded rectangle represents the activities. Diamonds are used to mark choices. Bars represent join or split of concurrent activities. The contents of the diagram may be organized into partitions called swim lanes which are marked with solid vertical line that divides the diagram into segments.

Below is a simplified example of activity diagram for processing an order. The diagram shows the flow of actions in the system. Once the order is received it will be filled, after it is filled the one responsible for shipping the order will verify the delivery method via choice of instant delivery or regular delivery. In the end both choices result in same activity of sending the order thus the diagram reaches final state.



©Jarkko Hurme

Picture 9. Simplified example of Activity Diagram

This information gathered from the activity diagram then can be used to create test cases similar to UML – state machine diagram. For example from the diagram we can see that we need a customer, order, product and employee tables to form the system.

Naturally these tables must contain the necessary information for them. Without seeing neither the system nor other diagrams about it I can deduce that the customer table will need at least; ID, Name, Address, Country, Postal Code for the delivery to reach the customer. If the system is an online shop Password and Username is necessary as well as other contact information like Phone number, Email.

The order table will need at least two different delivery options thus it could be done either with a radio button or a dropdown menu. This leads to the conclusion that a viable test case could be trying to place an order without selecting delivery option or selecting multiple delivery options.

Creating test cases with activity diagram is more challenging than using UML – state machine diagram as you need to visualize the whole system in your head with only the aid of activities but it is a viable option as well for creating test cases.

5 UML – BASED TESTING IN PRACTISE

The main principle on UML – based testing is to create test cases with the aid of equivalence classes and state machines. These created test cases are then injected into state machines which are then compared to the test results. If the values from the test cases can be verified, the test has succeeded.

El-Far (El-Far & Whittaker 2001, 6-8). has listed multiple resources for creating test cases for a system from which I adapted the ones that suited the purpose best. Below are the guidelines that I try to follow when visualising the system or when creating test cases about the system.

- Define the components and features to be tested. Defining what will be modelled is the first part of the process.

- Start to study the system, proceed to analyze the structure of the program and functionality of system while keeping an eye out for potential error situations.

- Collect relevant and useful documentation about the system. Testers need to get as much information about the system and its functions as possible.

- Discuss the system with co-workers/colleagues to get a better understanding of the system as well as agreeing on the tools that will be used.

- Identify the users and their inputs, to be able to react better to errors caused by false input.

- Inspect the interface. It is important to familiarize with the interface in order to recognize abnormal feedback from the system or errors in the interface.

- Explore the range of inputs. It is beneficial to divide the inputs into equivalence classes which will aid in creating test cases.

- Explore the suitability of feeds as well as their sequences. The model must contain valid information about the terms and conditions in the system before the test cases can be made. e.g. you cannot test a button in a form before you open the form.

5.1 From plan to practise

I first started my work on eTM by creating an activity diagram to get a better view of the system and to gain insight about the multiple roles and their purposes. The process was started by gathering information about the system. This information was gathered from three main sources:

1. Regiona Toscana made discussion guide about IDEAL – EU a European Town Meeting on climate change on November 15th 2008. This discussion guide was sent to the participants before the meeting.

From this guide I managed to gather information about Central Facilitator, Theme Teams, Participants as well as Tables.

2. Eyewitness statements of how the system works.

These contradicted a lot with the information acquired from the discussion guide e.g. there was no mention about Central Facilitator in the eyewitness statements. Other differences included floor managers, photographers and secretaries which were not mentioned in the discussion guide.

From this data I compiled that there are numerous possible setups to run an eTM and only certain key roles are mandatory. After researching all the information gathered so far I drafted the first version of my activity diagram. This version had multiple flaws and it was more of an activity diagram about the setup of a meeting than a diagram that describes the activities of the meeting itself though with this diagram I managed to get a better understanding of the processes prior to the meeting.

3. The eTM – software

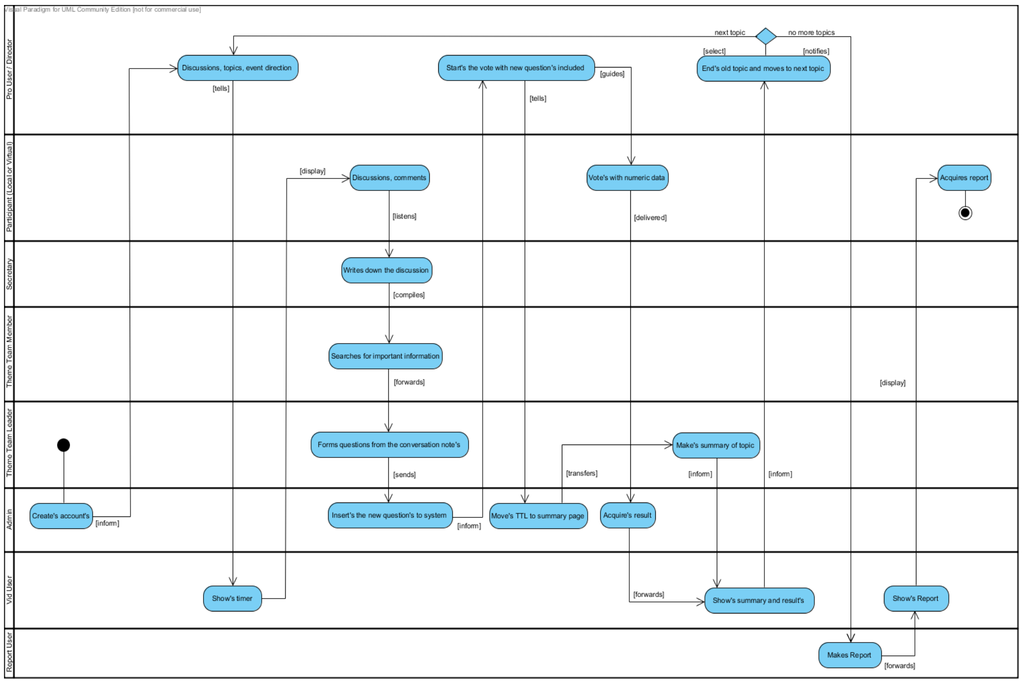
The final part of the puzzle came together when I saw the software for the first time. The system was not perfect and it had flaws. At that time, though by observing the interface and the source code I managed to gain a greater understanding of the software as well as find out the key roles. This was not however the only benefit that I gained from the software. It had offered me a guideline which to follow so I could weed out the contradictions and misleading information that had emerged from my previous sources.

The vertical swim lanes correspond to the roles found in eTM – software from top to bottom they are: Producer, Participant, Secretary, TTM, TTL, Admin, Video user and Report user. The lines show the flow of actions in the diagram from Admin creating the user accounts to end of eTM when Participants are handed the final report.

All this new insight about the system had made me realise another flaw in my diagram draft. This time it was a structural flaw although it did not affect the diagram as much as my previous flaw I had decided to start a new.

In my first draft I had used horizontal approach to draw the actions and this time I used vertical approach. This aided me when drawing actions as I could organize them better to avoid overlap of actions thus making the diagram a lot clearer to read.

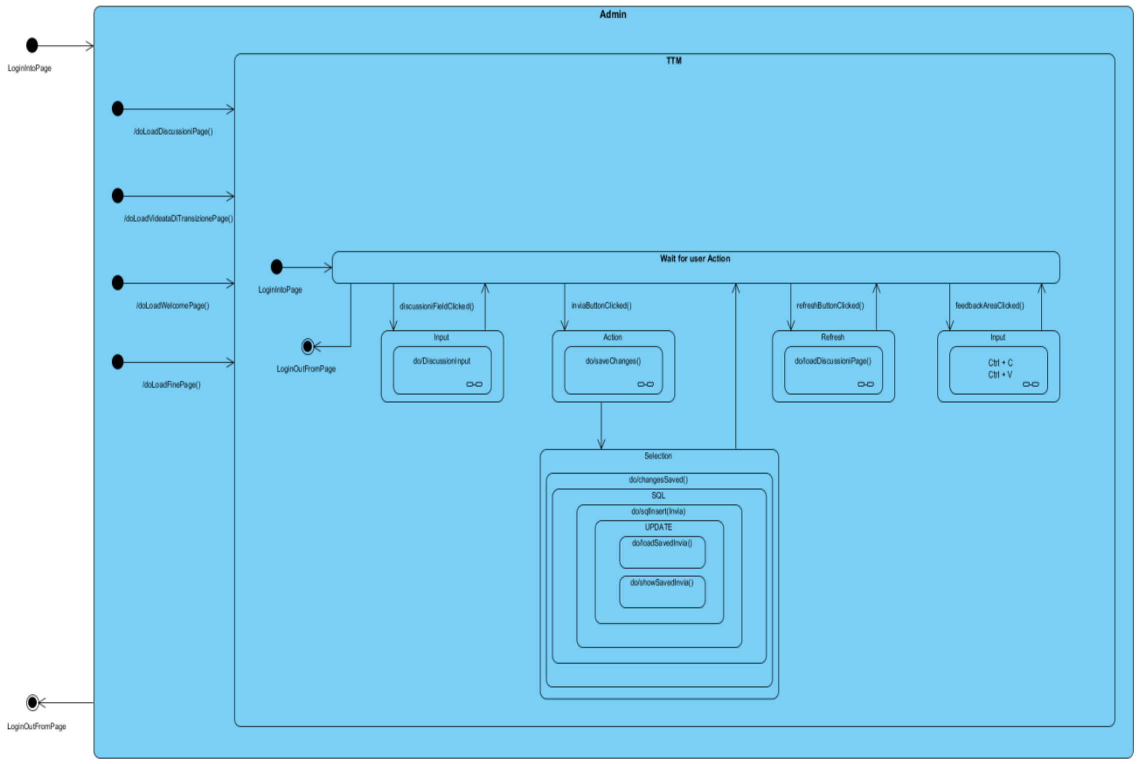
As the second version of my activity diagram was finished I had managed to familiarize myself well enough with the system to start designing UML – state machine diagrams.



Picture 10. eTM Activity Diagram

I began my work from one of less complicated roles, TTM. With the aid of the activity diagram and screenshots from the interface I began my work of drawing the UML – state machine. The TTM state machine consists of four states and one state which is not affected by TTM: s actions but is affected by the Admin. After I had finished the TTM state machine I was not content with the diagram something in it bothered me. In due time after creating TTL, VID, Report and Participant state machines, it was my instructor Oskari Kiviniemi who helped me to realize that it was the Admin state in the TTM – state machine that had bothered me.

Admin has the ability to transfer the user from screen to screen. That was the flaw in my TTM state machine, I had not realised that the TTM does not affect the process thus it cannot be marked as state for the TTM. The solution was to create a clear hierarchy to the UML – state machine diagrams which showed the Administrators state in different layer than the TTM's states.



Picture 11. Hierarchy of TTM – State machine

After finishing the necessary fixes to my already completed diagrams, I started to work on the Producer UML – state machine. This state machine was three times the size of the ones I had done so far. Due to its size and how the interface layout was I decided to divide it into three sections.

- Who Makes What, which is used to set the actions of the other roles by transferring their screens from one screen to another for example moving Participants to polling screen while TTL is kept on its main interface screen.
- Action Summary which is used to list what everyone is doing currently.
- Polling Panel Management, which can be used to start or stop polling sessions or to compare polling results or to insert new questions to the polls.

After the small state machines were complete I started to examine their relationships by drawing their connections to each other. This led me to notice that my Producer state machine was missing one tier of hierarchy which I fixed by inserting another state machine inside the Polling Panel Management. This new state machine called Vedi was then connected to Action Summary thus the relationships between the state machines were corrected. Finally I inserted these small state machines inside the Producer state machine thus creating the last layer of hierarchy that is present in the Producer UML – state machine.

Finishing Producer UML – state machine meant that I only had one more state machine to draw the Administrator state machine. Administrator state machine was the most complex and largest state machine of them all. Thus it is no wonder that I had my share of problems while doing it. In the end I decided to divide the diagram into four sectors similar to the Admin interface. All of these sectors included multiple sub state machines listed below.

These sectors were:

- Event Setting
  - Sectors Management
  - Users Management
  - Static texts translation
  - Assign tables to TTM
  - Assign TTM to TTL

- Event Management
  - Discussion session management
  - Polling session management

- Event Setting
  - Who does what
  - Actions Summary

    - Polling session

- Maintenance
    - Session Data Management

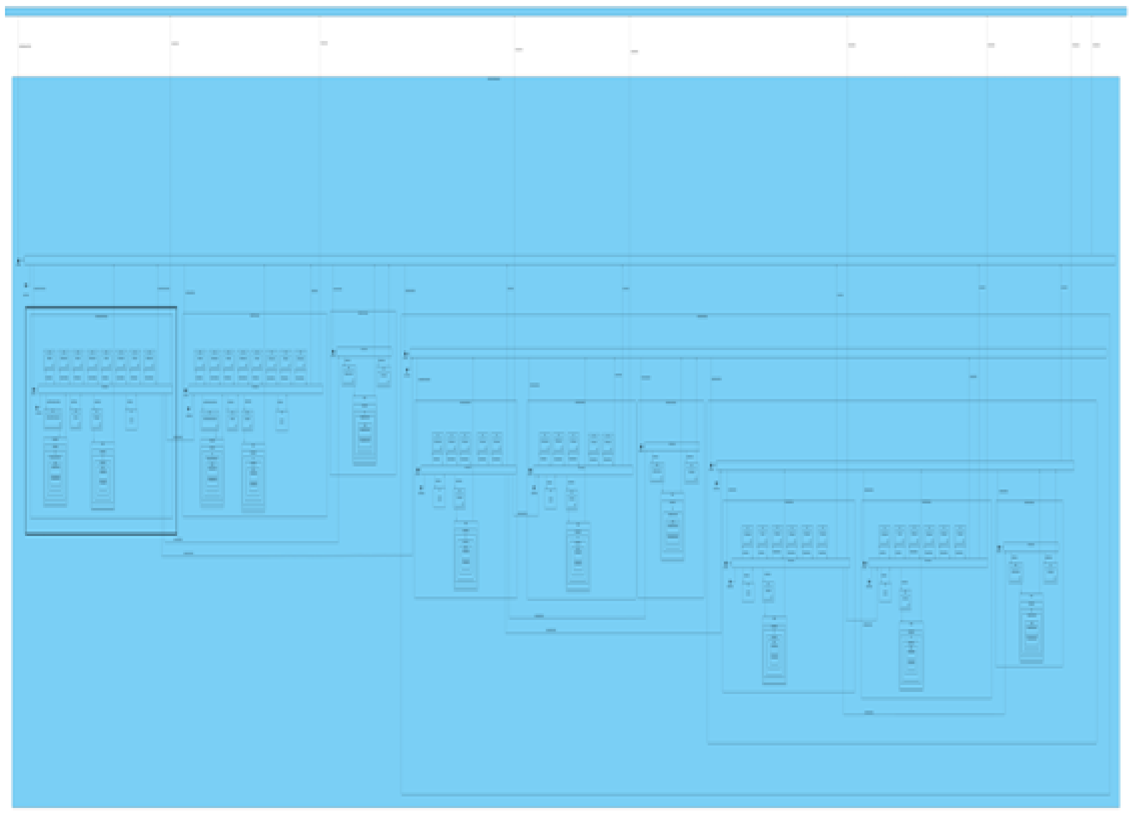
This allowed me to get a better grasp of the Admin state machine as a whole though I was still left with the problem of how to display the state machines due to their size. After a consultation with my supervisor I decided to build a separate state machine which aided me on displaying the transition between these massive state machines as well as acting as the initial state and final state to them.

After finishing the Administrator UML – state machine I finally started to mark down the equivalence classes of each state machine, which would make the process of creating test cases a lot simpler.

5.2 Creating test cases for eTM

This sub-chapter discusses the creating of test cases with UML – state machine and equivalence classes. I will use Polling Session Management from Administrator UML – state machine as the source for this demonstration.

First I show the picture of the whole Polling Session Management then I divide it to smaller more manageable sections. After describing this process I move to tell about the test case creation process.



Picture 12. Polling Session Management

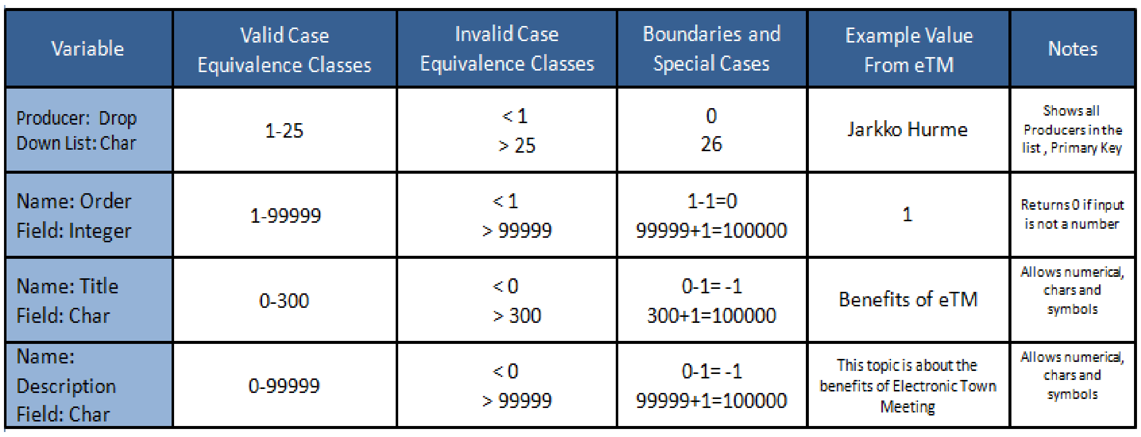
As shown in the above picture the Polling Session Management is large and complex state machine. Here I focus on one sub state machine marked with a black square. The selected diagram portion is described in detail in appendices.

The sub state machine selected is the New Polling Session, which is related to the key feature of eTM system. Without the New Polling Session it would be impossible to insert questions into the system – create answer options or to vote.

After the base is selected it is possible to look at all the available information on the New Polling Session sub state machine. Naturally there is the sub state machine itself as well as the equivalence classes drawn after finishing the Administrator state machine.

.

Figure 2. Equivalence classes related to the New Polling Session

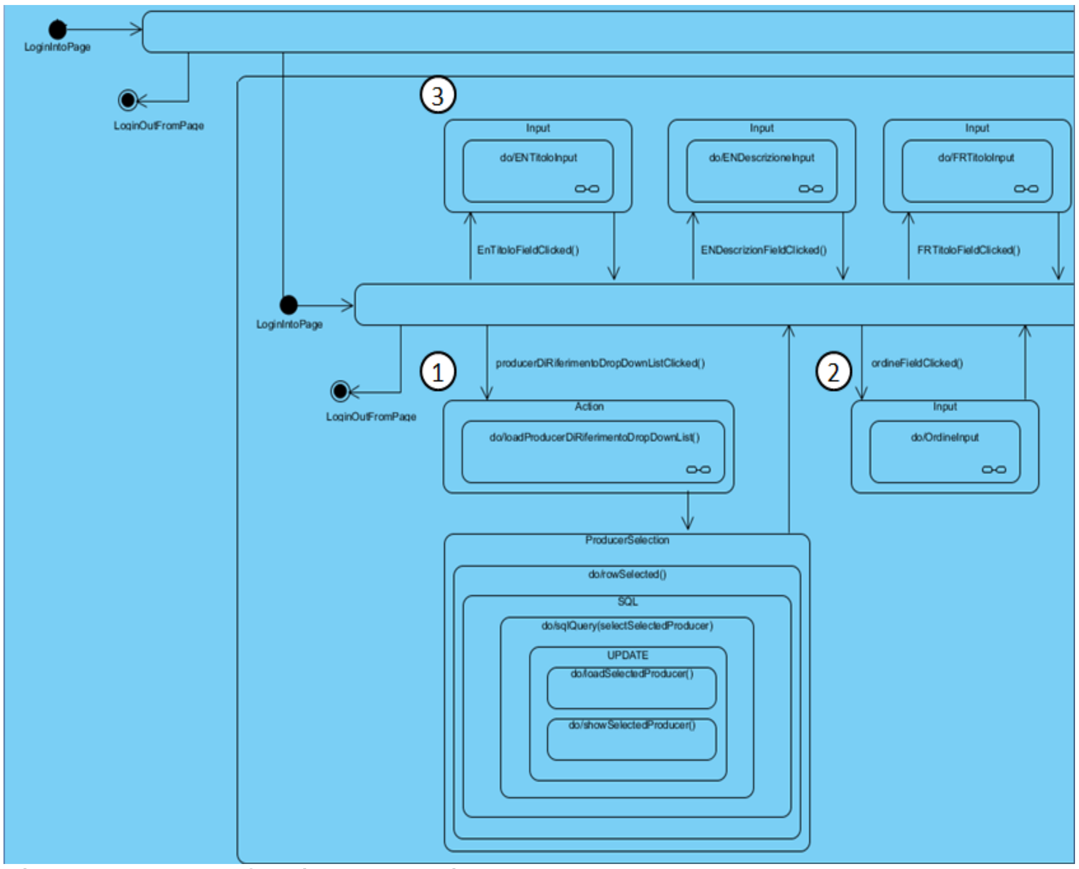


Variable	Valid Case Equivalence Classes	Invalid Case Equivalence Classes	Boundaries and Special Cases	Example Value From eTM	Notes
Producer: Drop Down List: Char	1-25	< 1 > 25	0 26	Jarkko Hurme	Shows all Producers in the list , Primary Key
Name: Order Field: Integer	1-99999	< 1 > 99999	1-1=0 99999+1=100000	1	Returns 0 if input is not a number
Name: Title Field: Char	0-300	< 0 > 300	0-1= -1 300+1=100000	Benefits of eTM	Allows numerical, chars and symbols
Name: Description Field: Char	0-99999	< 0 > 99999	0-1= -1 99999+1=100000	This topic is about the benefits of Electronic Town Meeting	Allows numerical, chars and symbols

By looking at the list of all equivalence classes related to this state machine, it is clear that majority of them are text fields. The values and notes show that there are not many restrictions related to the system and that the input for fields can be one char at minimum.

Naturally there will be situations when one cannot be certain e.g. about the valid case equivalence classes due to lack of information. In those situations is it best to estimate the value.

Now that the fields from the state machine and their equivalence classes are known, test cases based on the information can be created. The process is relatively simple: look at the state machine and its field’s, compare them to equivalence classes and compile that information into test cases. Next I demonstrate the creation of test cases with three states marked in the picture below.



Picture 13. Base for demonstration test cases

To start from the Producer selection; we know it’s a Dropdown - list and we know the equivalence classes. That is enough information to create the test cases mentioned next.

A. What if the Admin selects a specific Producer?

It is important to test that if it is possible to select a certain Producer from a list of 10/100/1000 Producers.

B. What if the Admin does not select a Producer?

There is a possibility that an eTM does not need or have a Producer, thus the option of not selecting one has to be tested.

C. What if the Admin tries to select multiple Producers?

There is a change that on eTM there will be numerous Producers, thus it is important to test if multiple Producers can be selected.

After the creation of test cases it is time to test the system and verify what the application does. Below is the list of test cases created and the feedback from the application.

These test cases are colour - coded to allow easier success/fail tracking as well as test case analysis. Green colour means Successful test case while red is failed test case. The test cases include a screenshot of the end result as well. This is included for the validity and reproducibility of the test cases.

Table 1. Test Case 1A

Test - ID	**1A**
Case Name	Selecting a Producer
Test Sequence	Selecting a Producer from dropdown -list
Purpose	Selecting a specific Producer from the dropdown - list

Covered Use Cases	1
Covered Equivalence Classes	1-25

Success End Condition	The user can select a specific Producer from the list
Failure End Condition	The user is not able to select a specific Producer

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Producer dropdown - list
	Tester selects specific Producer from the list

Stability Evaluation Condition	Tester selects specific Producer from the list of 10 people
	Tester selects specific Producer from the list of 100 people

Test Case Result	Successful
Stability Evaluation Result	Successful

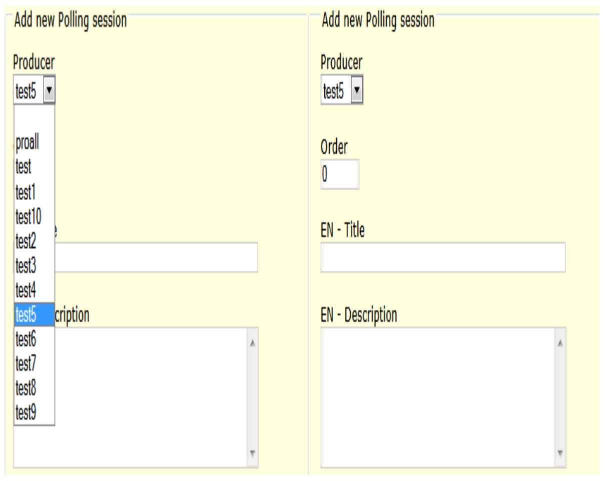
Screenshot	 Add new Polling session Producer test5 proall test test1 test10 test2 test3 test4 test5 test6 test7 test8 test9 cription Add new Polling session Producer test5 Order 0 EN - Title EN - Description

Table 2. Test Case 1B

Test - ID	**1B_1**
Case Name	No Producer
Test Sequence	Tester does not select a Producer
Purpose	Testing if Producer is mandatory

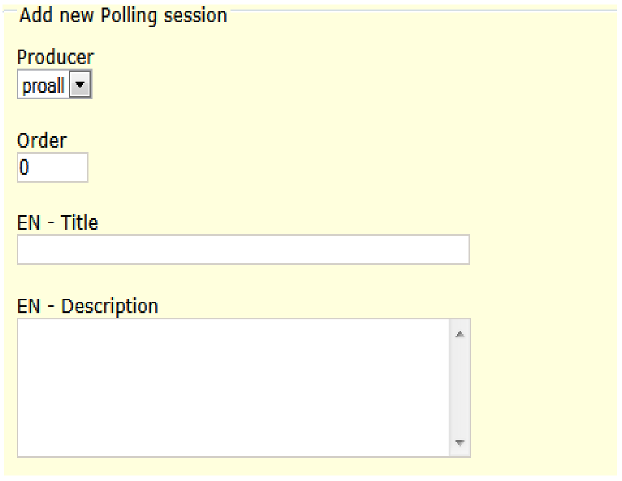
Covered Use Cases	2
Covered Equivalence Classes	1-25

Success End Condition	The user has not selected a Producer
Failure End Condition	The user is not able to leave the Producer field null

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester does not click the Producer dropdown - list

Test Case Result	Failure, selects automatically the first Producer as default

Screenshot	 Add new Polling session Producer: proall Order: 0 EN - Title EN - Description

Test - ID	**1B_2**
Case Name	No Producer
Test Sequence	Tester deletes all Producers then repeats 1B_1
Purpose	Testing if Producer is mandatory

Covered Use Cases	2
Covered Equivalence Classes	1-25

Success End Condition	The user has not selected a Producer
Failure End Condition	The user is not able to leave the Producer field null

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Users Management
	Tester deletes all Producers
	Tester clicks the Homepage
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester does not click the Producer dropdown - list

Test Case Result	Successful , if there are no Producers the field can be empty

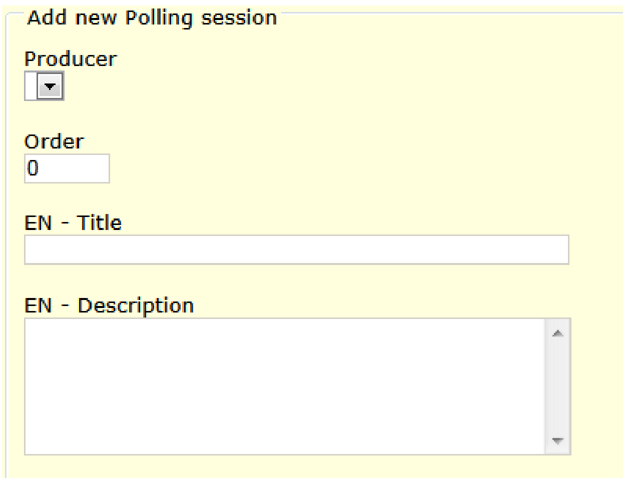
Screenshot	 Add new Polling session Producer Order 0 EN - Title EN - Description

Table 3. Test Case 1C

Test - ID	**1C**
Case Name	Multiple Producers
Test Sequence	Tester tries to select multiple Producers with Ctrl hold down.
Purpose	Testing if multiple Producers can be selected

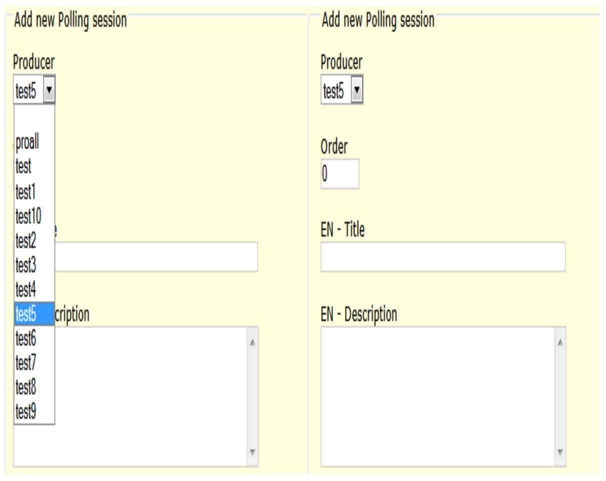
Covered Use Cases	1
Covered Equivalence Classes	1-25

Success End Condition	The user has selected multiple Producers
Failure End Condition	The user is not able to select multiple Producers

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Producer dropdown - list with Ctrl

Test Case Result	Failure, it is not possible to select multiple Producers.

Screenshot	

Next the Order field is tested. The process is same as with the Producer selection, the only difference is that it is an input field instead of a Dropdown - list. Possible test cases for order field can be.

A. What if Admin writes numbers on the field?

The correct input for the field, though naturally it must be tested as well to see that the system works.

B. What if Admin writes chars on field?

As shown in equivalence classes there are no constraints related to the field, thus it is possible to type other inputs than numbers on the field. That is why it has to be tested how they affect the system if they are accidentally used on the field.

C. What if the Admin writes symbols on the field?

It is important to test how different input types affect the system.

D. What if Admin does not type into the field?

There is a chance that the Admin does not remember to type an input to the field.

E. What if the input exceeds over the max field size?

It is important to test how the system reacts in case the field size is exceeded.

After creating a number of test cases for that state it is time to test the system again.

Table 4. Test Case 2A

Test - ID	**2A**
Case Name	Number Test
Test Sequence	Tester input's numbers on the field
Purpose	Purpose is to test if the field accepts numbers

Covered Use Cases	1
Covered Equivalence Classes	00001-99999

Success End Condition	The user can input numbers to the field.
Failure End Condition	The user is not able to input numbers.

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester types 123 on the field

Test Case Result	Successful

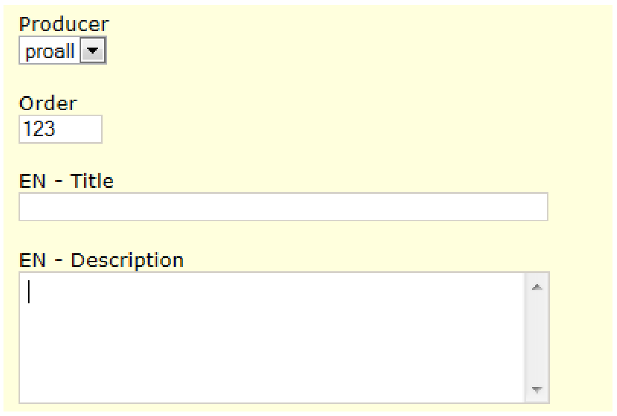
Screenshot	 Producer: proall; Order: 123; EN - Title; EN - Description

Table 5. Test Case 2B

Test - ID	**2B_1**
Case Name	Char Test
Test Sequence	Tester input's chars on the field
Purpose	Purpose is to test if the field accepts chars

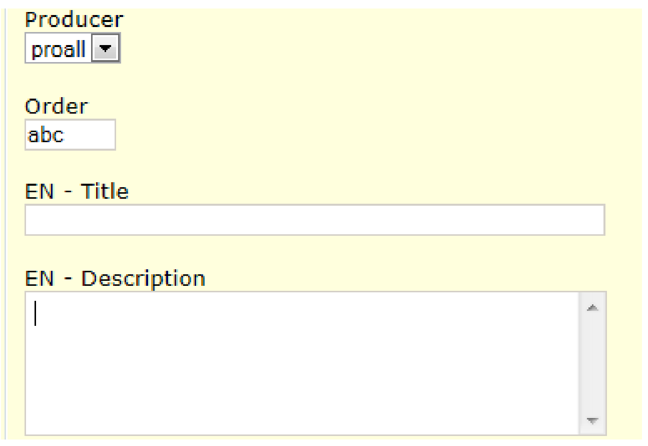
Covered Use Cases	2
Covered Equivalence Classes	1-99999

Success End Condition	The user is not able to input chars on the field.
Failure End Condition	The user can input chars to the field.

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester types abc on the field

Test Case Result	Failure, the field does not prevent char input

Screenshot	 Producer: proall Order: abc EN - Title EN - Description

Test - ID	**2B_2**
Case Name	Char Test
Test Sequence	Tester input's chars on the field and then saves
Purpose	Purpose is to test if the field accepts chars

Covered Use Cases	2
Covered Equivalence Classes	1-99999

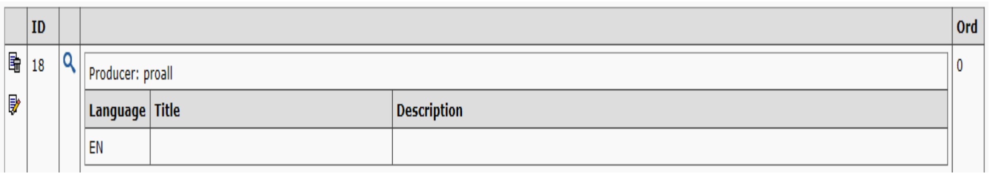
Success End Condition	The user is not able to input chars on the field.
Failure End Condition	The user can input chars to the field.

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester types abc on the field
	Tester presses Save
	Tester presses OK

Test Case Result	Successful , eTM does not allow saving chars in Order field

Screenshot



ID			Ord
18		Producer: proall	0

Language	Title	Description
EN		

Table 6. Test Case 2C

Test - ID	**2C_1**
Case Name	Symbol Test
Test Sequence	Tester input's symbols on the field
Purpose	Purpose is to test if the field accepts symbols

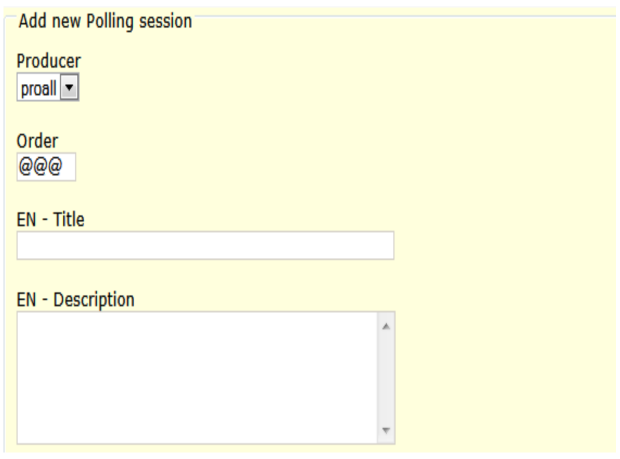
Covered Use Cases	2
Covered Equivalence Classes	1-99999

Success End Condition	The user is not able to input symbols on the field.
Failure End Condition	The user can input symbols to the field.

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester types @@@ on the field

Test Case Result	Failure, the field does not prevent symbol input

Screenshot	 Add new Polling session Producer proall Order @@@ EN - Title EN - Description

Test - ID	**2C_2**
Case Name	Symbol Test
Test Sequence	Tester input's symbols on the field
Purpose	Purpose is to test if the field accepts symbols

Covered Use Cases	2
Covered Equivalence Classes	1-99999

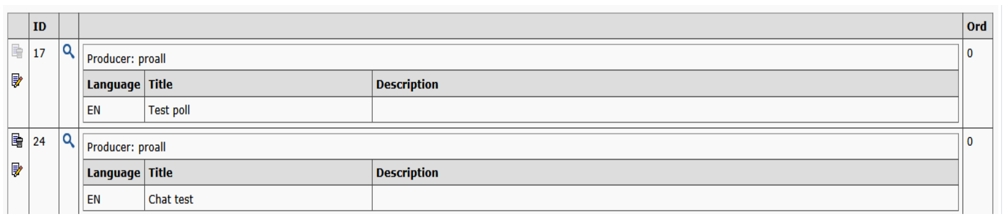
Success End Condition	The user is not able to input symbols on the field.
Failure End Condition	The user can input symbols to the field.

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester types @@@ on the field
	Tester presses Save
	Tester presses OK

Test Case Result	Successful, eTM does not allow saving symbols in Order field

Screenshot



ID			Ord
17		Producer: proall	0
		Language / Title / Description	
		EN / Test poll /	
24		Producer: proall	0
		Language / Title / Description	
		EN / Chat test /	

Table 7. Test Case 2D

Test - ID	**2D**
Case Name	Field Length Test
Test Sequence	Tester input's max amount of numbers into the field
Purpose	Purpose is to test if the field limit can be broken

Covered Use Cases	1
Covered Equivalence Classes	1-99999

Success End Condition	The user is not able to input more numbers than field allows
Failure End Condition	The user can input numbers over the max limit of the field

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester types max amount of numbers 99999 on the field

Test Case Result	Successful

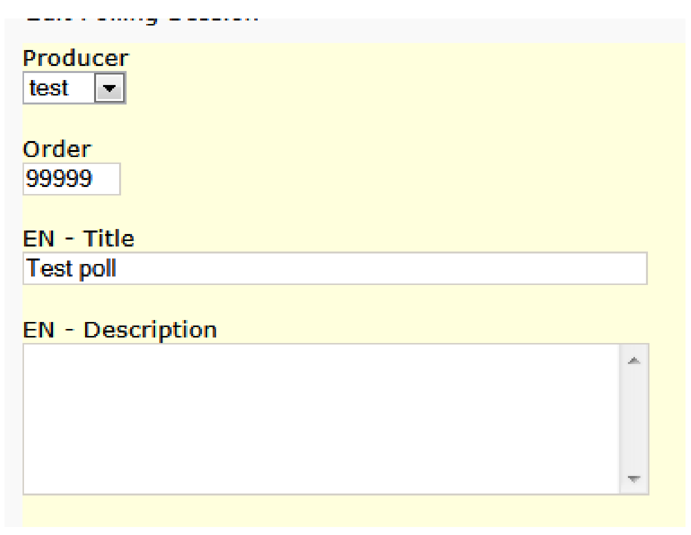
Screenshot	 Producer: test; Order: 99999; EN - Title: Test poll; EN - Description:

Table 8. Test Case 2E

Test - ID	**2E_1**
Case Name	Empty Field Test
Test Sequence	Tester does not input any value to the field
Purpose	Purpose is to test if the field can be left empty

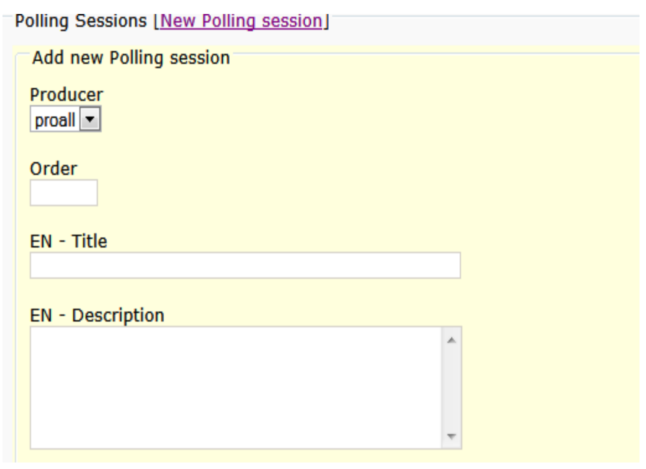
Covered Use Cases	2
Covered Equivalence Classes	1-99999

Success End Condition	The user can leave the field empty
Failure End Condition	The user is not able to leave the field empty

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester leaves the field empty

Test Case Result	Successful

Screenshot	

Polling Sessions [New Polling session]

Add new Polling session

Producer: proall

Order

EN - Title

EN - Description

Test - ID	**2E_2**
Case Name	Empty Field Test
Test Sequence	Tester does not input any value to the field
Purpose	Purpose is to test if the field can be left empty

Covered Use Cases	2
Covered Equivalence Classes	1-99999

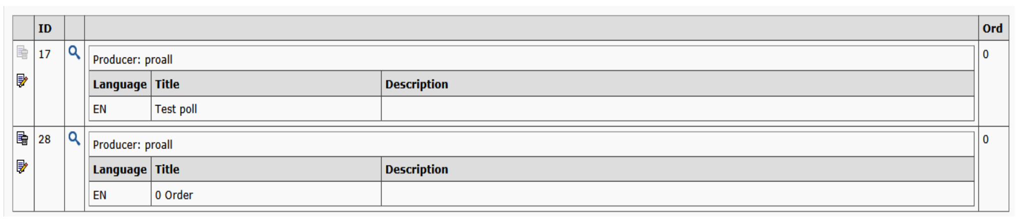
Success End Condition	The user can leave the field empty
Failure End Condition	The user is not able to leave the field empty

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the Order field
	Tester leaves the field empty
	Tester presses Save
	Tester presses OK

Test Case Result	Failure, eTM returns the default value to the field, 0

Screenshot



	ID			Ord
	17		Producer: proall	0
			Language | Title | Description	
			EN | Test poll |	
	28		Producer: proall	0
			Language | Title | Description	
			EN | 0 Order |	

In this demonstration I show the result immediately after creation of test cases. Depending on situation it is not necessary to start the testing process immediately after finishing a state but to run all the test cases after finishing test case creation for the whole state machine.

Naturally the same test cases used for Order field can be run to test that title field work as intended, though new test cases that are specific for the title field can also be created.

.

A. What if Admin types only in the second title field?

It is possible that eTM – software has numerous active languages while only one is needed for conference. That is why it is important to find out if it is possible to fill a title field in English but leave it empty in Italian.

B. What if Admin writes on both fields?

In certain situations the need for multiple languages might rise. That is why it should be tested if it possible to display as well as write to both title fields.

C. What if both title fields are left empty?

In certain scenario there may be no need for title, thus it is important to test if the New Polling Session – part of the system can operate without a title.

Table 9. Test Case 3A

Test - ID	**3A**
Case Name	One Title
Test Sequence	Tester input's chars on the EN title but leaves IT title empty
Purpose	To test if the system allows only use of one language when there are two active languages

Covered Use Cases	1
Covered Equivalence Classes	0-300

Success End Condition	The user can type in EN field and leave IT field empty
Failure End Condition	The user is not able to leave IT field empty

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the EN title field
	Tester types test on the field
	Tester ignores the IT title field
	Tester presses Save
	Tester presses OK

Test Case Result	Successful

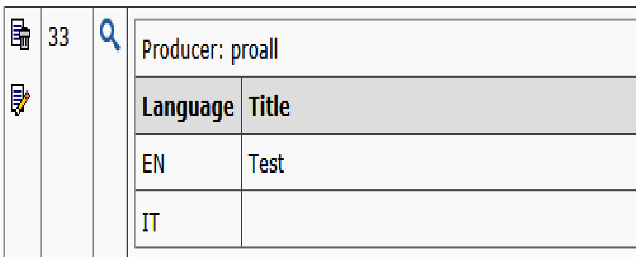
Screenshot	 33 Producer: proall Language | Title EN | Test IT |

Table 10. Test Case 3B

Test - ID	**3B**
Case Name	Two Titles
Test Sequence	Tester input's chars on the EN title and IT title field
Purpose	To test if the system allows use of two title fields when there are two active languages

Covered Use Cases	1
Covered Equivalence Classes	0-300

Success End Condition	The user can type in EN title field and IT title field
Failure End Condition	The user is not able to type in both fields

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester clicks the EN title field
	Tester types test on the field
	Tester clicks the IT title field
	Tester types test on the field
	Tester presses Save
	Tester presses OK

Test Case Result	Successful

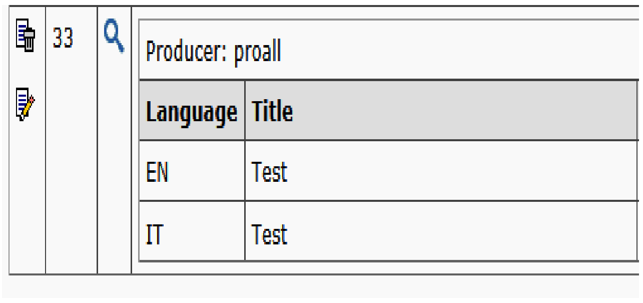
Screenshot	 33 Producer: proall Language | Title EN | Test IT | Test

Table 11. Test Case 3C

Test - ID	**3C**
Case Name	No title
Test Sequence	Tester does not input a title
Purpose	Purpose is to test if title is mandatory

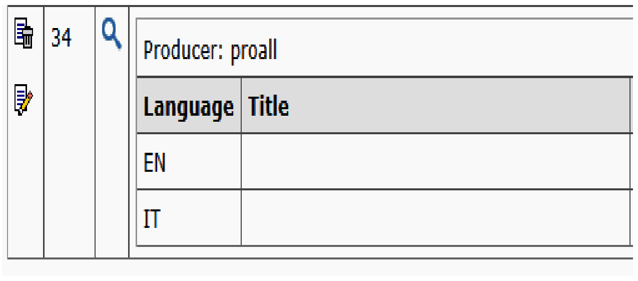
Covered Use Cases	1
Covered Equivalence Classes	0-300

Success End Condition	The user is able to leave the field empty
Failure End Condition	The user is not able to leave the field empty

Preconditions	Steps
	Tester logs into eTM as Admin
	Tester clicks the Polling Session Management
	Tester clicks the New Polling Session

Test Case Input	Steps
	Tester presses Save
	Tester presses OK

Test Case Result	Successful

Screenshot	

34

Producer: proall

Language	Title
EN	
IT	

This final part of the demonstration shows that it is possible to invent new test cases even when it seems that the fields are identical.

It should also be noted to avoid testing all possible options of certain test cases. This test case included two languages of possible five, thus we can assume that the function will works with all five selected as active.

That means that there is no need to spend resources to perform the same test with different amount of options which results that those resources can be spend to test another function in the system.

6 CONCLUSIONS

Testing has a significant role in software development. Existing software still needs to be further developed and new software has to be released constantly. Testing is an essential part of reaching that goal.

The focus of this thesis was to concentrate on the benefits of UML – tools in the testing and evaluation of eTM – software. The goal was to give the reader a good impression of the topic in hand.

First the theory about eTM – software as well as UML was explained in chapters 2 and 3, to help the reader understand the frame for the tests and to familiarize them with the tools that are used. In chapter 4 of the thesis I concentrate on using UML – tools in testing to give comprehensive and practical examples of how to use these tools in software testing. Chapter 5 deals with the writers own experiences of testing the eTM – software with UML – diagrams.

It should be noted that this thesis is only a brief demonstration about the usability and benefits of injecting test cases in to the created UML – diagrams. There are numerous other software that could be tested in similar fashion and achieve as good results, than with other testing methods. It is also important to remember that the method mentioned in this thesis is still new and it will take time for us to fully understand what can be achieved with it.

In this thesis the testing process was used to describe the functionalities of eTM – software as well as to evaluate it by using UML State machine diagrams and Activity diagram to create test cases. With the help of the results received from the testing process the state of the software was evaluated.

The test cases were executed in multiple software versions of eTM. The expected results were compared to the actual results to verify if the tests were successful after this the found results and errors were reported to the appropriate personnel. While the test cases were performed attention was not only directed to them but to the evaluation of eTM as well by paying attention to the performance, availability and reliability of the system.

The origin of this thesis dates back two years ago. It was then, that I first started to ponder about a thesis subject which I would enjoy writing about. In the end I decided to create a website for a friend of mine and evaluate it as my subject.

This plan came to an end when I was offered a position in Living Lab – project. In this project I acted as a main software tester as well as developer; designing new functions to the original software created by Sauli Suominen and Ilari Hurme. (Hurme & Suominen, 2010.)

As time passed I found a passion for software testing. It was partially due Software Deployment & Testing – course (which was taught by Oskari Kiviniemi) as well as finding my own style for testing software. This was mainly because of the foreign exchange students from University of Lille. They provided me with feedback as we worked together on Living Lab – project which I then used to improve myself as a tester.

This led me to change my original thesis subject. I wanted to create my thesis about software testing. Soon afterwards I was offered a post as student assistant as well as a new project to join. That project was Parterre.

In the end I wanted to link my past with my future and thus my thesis subject had emerged. It was one which combined all my previous subjects together as well as my “future” project.

This whole process has been an intriguing experience. I have enjoyed writing this thesis and it feels like it has been only few days since I started it although it has been over two months now. In my opinion I have managed to fill the goals that I set for myself by creating an easy to read, compact and good information packet about the benefits of UML – diagrams in testing.

SOURCE MATERIAL

At1Ce. Referred 16.10.2011

http://www.at1ce.org/themenreihe.p?c=Software design

Booch, S.; Jacobson, I. & Rumbaugh, J. 1998. Unified Modeling Language User Guide, The. First Edition. Massachusetts: Addison Wesley.

Deek, F. & Noor, B. On the Design and Development of a UML - based Visual Environment for Novice Programmers. Journal of Information Technology Education Vol. 5. 2006.

El- Far, I. & Whittaker, J. 2001. Model – Based Software Testing. Florida Institute of Technology.

Regione Toscana. 2008. IDEAL - EU a European Town Meeting on climate change. Final Report.

Regiona Toscana. 2011. Un Town Meeting per la formazione del piano regionale di sviluppo economico. Guida Alla Discussione.

History of UML. Referred 13.10.2011

http://www.findthatfile.com/-/u/UML+History

Hurme, I.; Suominen, S. 2010. Projektienhallintasovelluksen toteutus ja testaus Living Labille. Opinnäytetyö. Tietojenkäsittelyn koulutusohjelma. Turku: Turun ammattikorkeakoulu, Salon yksikkö.

IBM Rational UML Resource Center. Referred 05.10.2011

http://www-01.ibm.com/software/rational/uml/

OMG. Referred 13.10.2011

http://www.omg.org/gettingstarted/what_is_uml.htm

Parterre. 2011. Parterre: Electronic Participation tools for spatial planning and territorial development. Booklet.

Parterre - Project. Referred 05.10.2011

http://www.parterre-project.eu/?electronic_town_meeting=1

Planning Tools Exchange. Referred 06.10.2011

http://www.planningtoolexchange.org/tool/electronic-town-meetings

Rebuilding New Orleans. Referred 16.10.2011

http://americaspeaks.org/wp-content/_data/n_0001/resources/live/UNOP_CaseStudy-LR.pdf

Sourcemaking. Referred 20.10.2011

http://sourcemaking.com/uml/basic-principles-and-background/history-of-uml-methods-and-notations

SparxSystems. Referred 20.10.2011

http://www.sparxsystems.com/enterprise_architect_user_guide/modeling_languages/profile_diagram.html

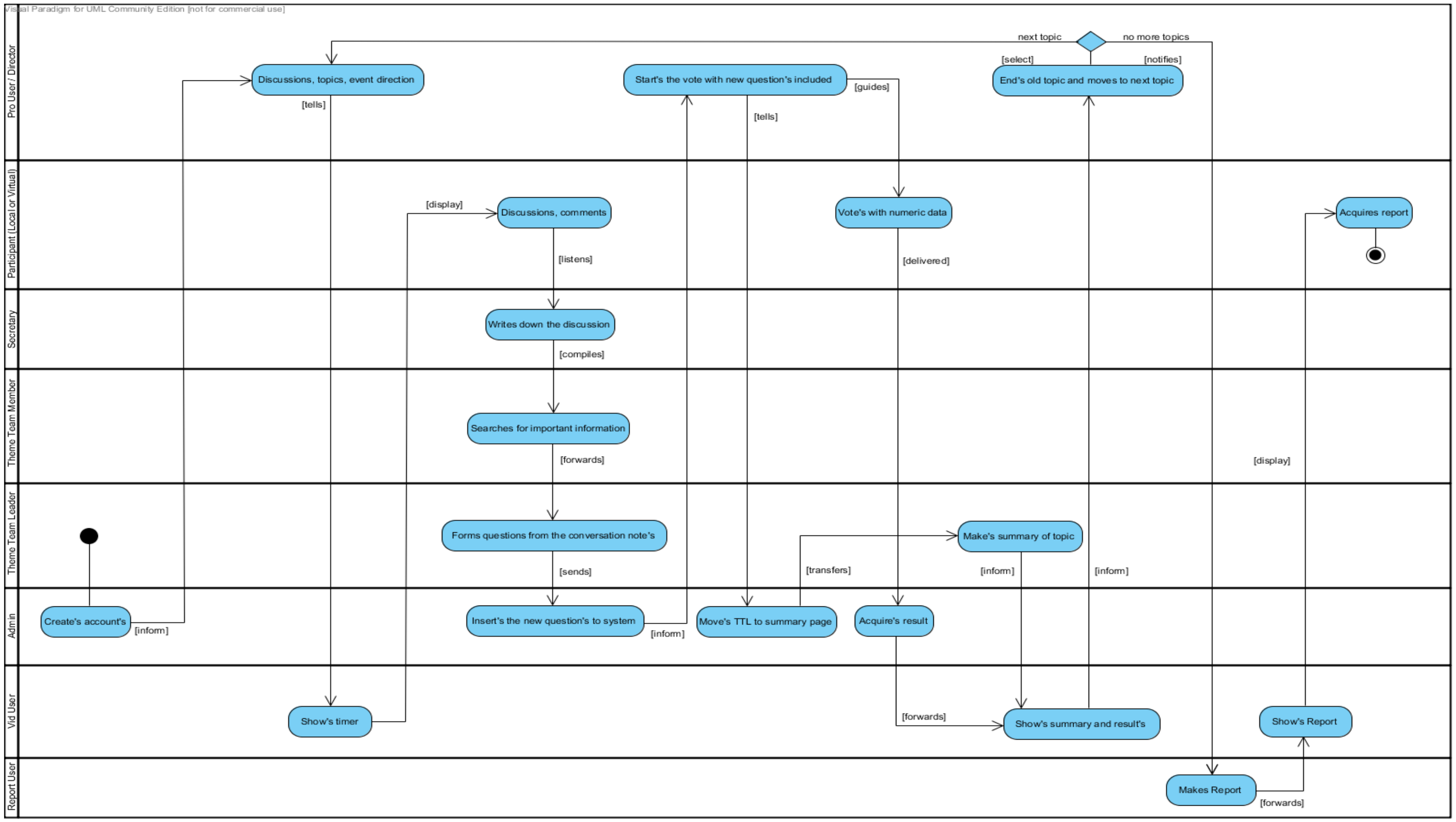
Wikipedia. Referred 16.10.2011

http://en.wikipedia.org/wiki/Main_Page

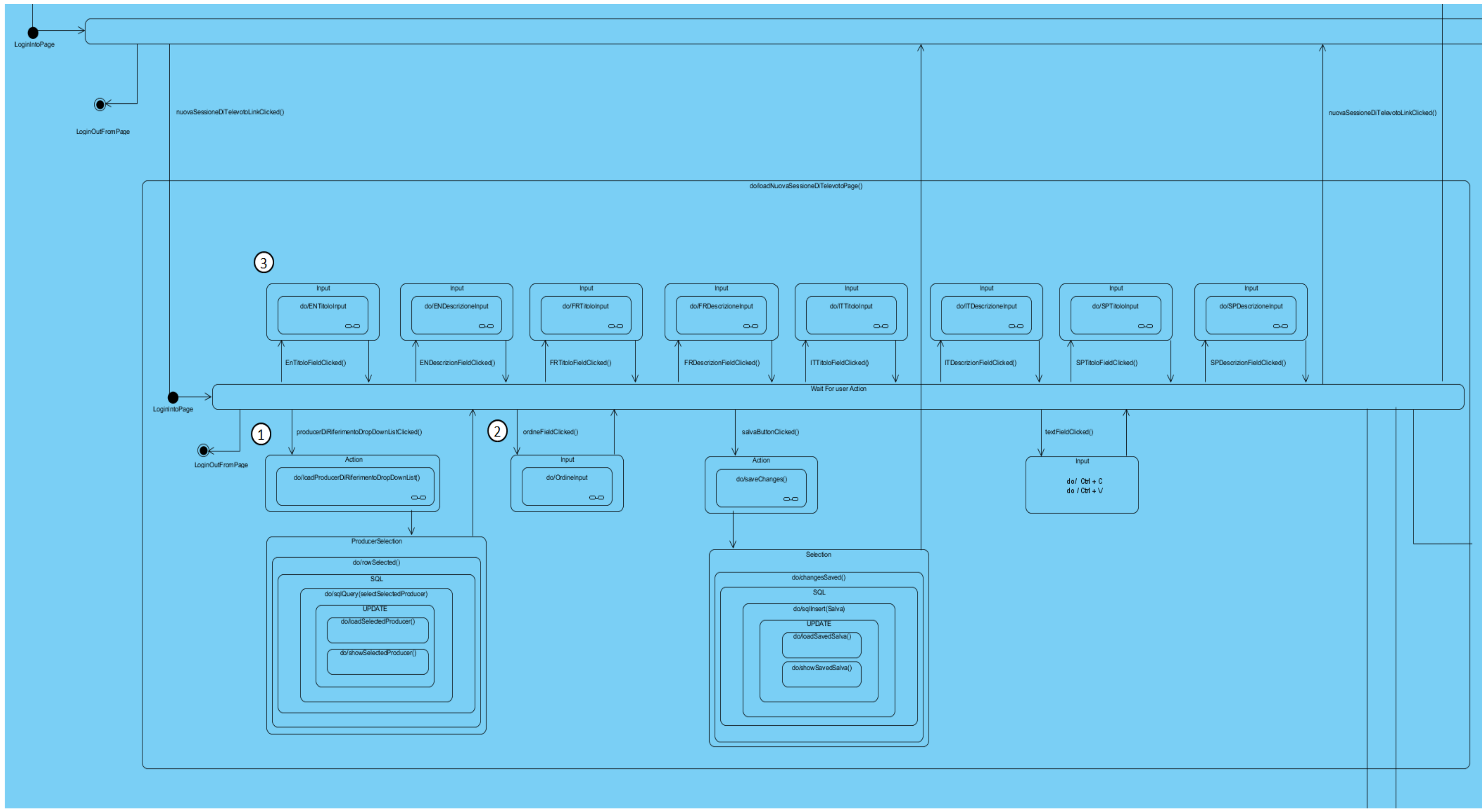
Xpdian. Referred 20.10.2011

http://www.xpdian.com/WhatistheUML.html

eTM Activity Diagram



New Polling Session


LoginIntoPage
LoginOutFromPage
nuovaSessioneDiTelevotoLinkClicked()
nuovaSessioneDiTelevotoLinkClicked()
do/loadNuovaSessioneDiTelevotoPage()
3
Input
do/ENTitoloInput
Input
do/ENDescrizioneInput
Input
do/FRTitoloInput
Input
do/FRDescrizioneInput
Input
do/ITTitoloInput
Input
do/ITDescrizioneInput
Input
do/SPTitoloInput
Input
do/SPDescrizioneInput
EnTitoloFieldClicked()
ENDescrizionFieldClicked()
FRTitoloFieldClicked()
FRDescrizionFieldClicked()
ITTitoloFieldClicked()
ITDescrizionFieldClicked()
SPTitoloFieldClicked()
SPDescrizionFieldClicked()
Wait For user Action
LoginIntoPage
LoginOutFromPage
1
producerDiRiferimentoDropDownListClicked()
Action
do/loadProducerDiRiferimentoDropDownList()
ProducerSelection
do/rowSelected()
SQL
do/sqlQuery(selectSelectedProducer)
UPDATE
do/loadSelectedProducer()
do/showSelectedProducer()
2
ordineFieldClicked()
Input
do/OrdineInput
salvaButtonClicked()
Action
do/saveChanges()
Selection
do/changesSaved()
SQL
do/sqlInsert(Salva)
UPDATE
do/loadSavedSalva()
do/showSavedSalva()
textFieldClicked()
Input
do/ Ctrl + C
do / Ctrl + V