Visual Modeling for Test Idea Generation

Vipul Kocher
Some Things That I Do
This is What THIS is All About

• Introduction
• A few types of models
• Tools for modeling
• Language based analysis
• Creating a new type of Visual model
  • Language of the model
  • Modeling a small problem
  • Using this model for designing tests
  • Extending the model
Models

• What are Models?
  • Structural design – MW dictionary
  • A simplified description, especially a mathematical one, of a system or process, to assist calculations and predictions – Oxford dictionary
  • a system of postulates, data, and inferences presented as a mathematical description of an entity or state of affairs A usually miniature representation of something - MW dictionary
  • A description or analogy used to help visualize something (as an atom) that cannot be directly observed - MW dictionary
Examples

• Architecture model – Android****
  **** from Android documentation

• ER Diagram - jcsites.juniata.edu

• Use-case diagram**
  **Source – Wikipedia

• Agile testing quadrants – informal model*
  * http://gojko.net/2013/10/21/lets-break-the-agile-testing-quadrants/
Why models?

• Words written or spoken are linear, Images are not!
Uses of Models

- To understand something complex
- To simplify something in order to communicate
- To use as an input for something else
- To use as a tool to reach a goal
Misuses of Models

• When model becomes an end goal
• When models are more complex than the problem you are trying to solve
• When a representation/model is wrong for a given stake holder
• When models are used to prove the reality wrong…
Unified Modeling Language (UML)

## UML Diagrams

<table>
<thead>
<tr>
<th>Diagram</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class diagram</td>
<td>Classes and their relationships</td>
</tr>
<tr>
<td>Component diagram</td>
<td>Components, their relationships and interactions</td>
</tr>
<tr>
<td>Composite structure diagram</td>
<td>Internal structure of classes/components/use-cases</td>
</tr>
<tr>
<td>Deployment diagram</td>
<td>Architecture of the system – what is deployed where</td>
</tr>
<tr>
<td>Object diagram</td>
<td>Objects and their relationships</td>
</tr>
<tr>
<td>Package diagram</td>
<td>Packages and the dependencies between them</td>
</tr>
<tr>
<td>Activity Diagram</td>
<td>High level business processes, data flow, logic</td>
</tr>
<tr>
<td>Use Case Diagram</td>
<td>Use-cases and their relationships</td>
</tr>
<tr>
<td>State Machine Diagram</td>
<td>State transition diagrams</td>
</tr>
<tr>
<td>Communication Diagram</td>
<td>Message flows between classes</td>
</tr>
<tr>
<td>Interaction Overview Diagram</td>
<td>Overview of the control flow</td>
</tr>
<tr>
<td>Sequence Diagram</td>
<td>Time ordering of messages between classes etc.</td>
</tr>
<tr>
<td>Timing Diagram</td>
<td>Change in state/condition/role over time</td>
</tr>
</tbody>
</table>
Context-free Questions

http://www.developsense.com/blog/2010/11/context-free-questions-for-testing/

http://enjoytesting.blogspot.in/2013/02/context-free-questions-mindmap.html
Tools for Modeling

• Old world
  • Pencil and paper
  • Whiteboard
  • Mathematics
  • …

• Software
  • Visio
  • Diagramming tools
  • Mind-mapping
  • IDE based tools
  • Excel
  • …
Language Based Analysis
Some software systems exist to manipulate specific kinds of data:
- The kinds of data sets are the nouns.
- The manipulations are the verbs.
- Adjectives and adverbs affect the manipulation.

Identify these items:
- Write test in sentences.
- Even trace coverage.

For example, an ATM system should:
- Accept deposits.
- Process withdrawals.
- Answer inquiries.
- Kinds of deposits, withdrawals, inquiries?
- Quickly? Securely?

This test design technique comes from Elisabeth Hendrickson.
Extending the Technique

• Extract the nouns and verbs
• Use 5W1H to extract adjectives, adverbs and more information
• Use pre and post conditions to extract more nouns and verbs
• Enumerate properties of every noun
• Break down these properties to base elements
• Use boundary values, equivalence classes on the base elements
• Write tests for individual properties
• Combine operations (verbs) to write scenario based test cases of arbitrary complexity
• Create test cases using pre and post conditions
Example: File Open

• Requirement fragment:
  • The <given> application allows users to open supported files.

• With the given requirement fragment
  • Very few meaningful test cases can be written using any test design technique
  • Using out extension to noun & verb technique we plan to write extensive test cases
Extracting the Nouns and Verbs

- The <given> application allows users to open supported files
- Extracting Nouns
- Extracting Verbs
- Extracting Adjectives and Adverbs
File open…

• Noun
  • Look for properties of this noun
  • Also ask What, Why, When, Where, Who, Which, How, How much/many

• Verb
  • Look for properties of this verb
  • Also ask What, Why, When, Where, Who, Which, How, How much/many
File: Properties

• Enumerate all the properties of FILE
• File has
  • Name
  • Size
  • Location
  • Extension
  • Metadata
    • Creation date
    • Accessing date
    • Modification date
• State
File: Properties

- Type
- Access Control
- Password protection
- Attributes
- Compression Support
- Encryption Support
- Creating application version
Breaking Down the Properties: Name

- Name - is a string
- String - is a SEQUENCE of Characters
  - Sequence = Length of string
  - Characters are elements of some valid character set
- (THUS) Name
  - Has a length
  - Has a set of allowed characters
  - And some characters are not allowed (/ or \ in file name on windows, spaces on Linux/Unix platforms)
Applying Boundary Values

• Name/String
  • Has a length (maximum and minimum)
  • Has a set of allowed characters (valid equivalence class)
  • And some characters are not allowed (invalid equivalence class)
Properties: Location

• Location
  • Is a string
    • Has a length (maximum and minimum)
    • Has a set of allowed characters (valid equivalence class)
    • And some characters are not allowed (invalid equivalence class)

• Also denotes physical location
  • On a locally connected drive (HDD, CD, Floppy…)
  • On the LAN
    • Using UNC path
    • Other operating systems (e.g. Linux connected using samba)
  • On a web page as a link
Properties: State

• State
  • in use/not in use
  • Saved/unsaved

• In use/open
  • Unmodified
  • Modified
  • New

• New/Modified
  • Without data
  • With data
Properties: Type

• Type:
  • Supported/not supported by the application
  • Binary or Text
  • Valid/invalid (corrupt)
Properties: Extension

• With extension
• Without extension

• Correct extension
• Incorrect extension
Properties: Attributes

• Read only
• Writable
• Archive
• Hidden

• ...

Dealing with the Verb(s)

• Applying 5W1H

• Open What?
  • File with various properties
  • Supported and unsupported files
  • ...

• Open Where?
  • Supported Operating Systems
    • Application under test
    • Browser plug-ins, if supported
Dealing with the Verb(s)…

• Open how?
  • Open in various ways
    • double click
    • drag and drop
    • command line
    • application menu
    • open with
    • ...

• Open how many?
  • Open one (minimum number of) file
  • Many files
    • Max. number of files that can be opened simultaneously…

• How fast
  • How much time does it take to open the file?
Some Simple Positive Test Cases

- Open a (supported) file
  - Minimum size
  - Maximum size
  - Minimum name length
  - Maximum name length
  - Located on a network drive/web
  - With and without appropriate extension
  - Created using older/newer version of creating application
  - Password protected
  - Read-only
  - …
Some Simple Negative Tests

• Open file(s) of
  • Unsupported type
  • Changed extensions
  • Corrupt file
  • Invalid/non-existent name
  • …
Scenario Based Tests

- Open a new file. Open another existing but closed file.
- Open an already open file.
- Open an existing file. Modify it. Open the same file again.
- Open an existing file. Modify it. Open another file.
- Open a local file. Open another file on the web.
- Open multiple files together. Check performance
- Open a very large file on the network. Check performance.
- Repeat the Open operation a number of times (repetitive tests for resource leaks…)
Pre-Conditions

• Before the operation is performed on the nouns think what conditions should be satisfied?

• For example, before file can be opened
  • File should exists
  • Application to open the file should be installed
  • Activation of the software is required?
  • Permissions need to be checked?
  • Preprocessing needs to be done?
  • Anything else?
Post-Conditions

• OK, I have performed the action. Now what?
Expanding the requirement

• What (after file open)?
• File is opened
  • Where
    • in the application window
      • What are properties of application window?
        • Location on the screen
        • Size of the window
        • State of GUI elements such as menus/buttons (enabling/disabling etc.)
    • What are the properties of OPEN FILE?
      • Cursor location
      • Zoom level
      • …
Writing more Tests

• What is the position of application once file is opened
• What is the cursor position after opening the file
• At what zoom level does the file open?
• …
Food for Thought

• A string needs to be displayed
  • What is the encoding?
  • What is the font for display?
    • Default font?
    • Size?
    • Color?
    • ...

Family of Nouns

• Are Nouns related to each other?
Mapping Relations

• Check if a noun is being used at multiple places
  • Is it referenced using the same name? (Syntax)
  • Does it mean the same at all other places? (Semantics)
  • Does it behave in the same manner at all other places? (Behavior)
• Are properties of one noun related to properties of other noun?
• Are there any constraints on the relations of the nouns.
• Write test cases on the basis of relations of nouns, relations of properties of nouns and constraints of the link
  • Example – If Age < 5 then no charges for movie ticket 😊

• Let us look at Insert Table feature of MS word
Insert Table

The image shows a screenshot of a software interface for inserting a table. The interface includes options for selecting the number of columns and rows, setting the AutoFit behavior, and choosing a table style. The table size can be customized to fit the desired dimensions.
Investigating Relationships

- Where is the width of each row set?
- Where is page width set?
- Is the unit of measurement constrained to be Inches?

Page Height = Rows * Width of each row

Page Width
Table Width = No. Column * size
Investigating Page Size - Margins
Investigating Page Size – Paper

Page Setup

Paper size:
- Letter
- Width: 8.5"
- Height: 11"

Paper source
- First page:
  - Default tray (Automatically)
  - Automatically Select
  - Main tray
  - Photo Tray
- Other pages:
  - Default tray (Automatically)
  - Automatically Select
  - Main tray
  - Photo Tray

Preview
- Apply to:
  - Whole document

Print Options...
Investigating Page Size – Layout
Unit Of Measurement

The image shows a software options window with a section titled "Measurement units." The available options include Inches, Centimeters, Millimeters, Points, and Picas. The selected option is "Inches."
Are Some Things missing?

• Where do I define the width of each row?
• What is the default?
• Can I change it? Where?
The Sanskrit grammar system described by Panini assigns karakas to verbal arguments based on the relationship they have with the verb.

Six Cases in Jainism

• That which independently (unaidedly) does (performs) its own deed (function) is called the doer (agent or Karta)
• Whatever modification is obtained by the subject (the doer) is called its function (Karma)
• The substantial cause of that particular deed by which it is done or originated is called the means of that deed (the Karan)
• That for which that particular deed is performed or done, is called the receiver (Sampradan karak)
• The permanent substance out of which that particular function or deed is done or obtained is called the (Apadan)
• The permanent cause and the same permanent substance is called the base of the deed (Adhikaran).
• Two types of cases –
  • Real - Achievement is said to be due to the substance itself
  • Conventional - When the achievement of the deed is shown as due to other agencies

http://www.jainworld.com/education/tatvagyan2/lesson06.htm
Applying Conventional Case

- The pot-maker is the doer
- The jar is the deed
- Wheel, stick etc. are means
- The pot-maker makes the jar for somebody to keep water in it, that somebody is the (Sampradan) or the receiver;
- The earth is taken out of the pit which is the Apadan (or the permanent cause)
- The pot is made on the base of the earth, which is the base or (Adhikaran)
... If a customer enters a store with the intention of buying a toy for a child, then advice must be available within a reasonable time concerning the suitability of the toy for the child. This will depend on the age range of the child and the attributes of the toy. If the toy is a dangerous item, then it is unsuitable. ...

<table>
<thead>
<tr>
<th>Part of speech</th>
<th>Model component</th>
<th>Example from the text</th>
</tr>
</thead>
<tbody>
<tr>
<td>proper noun</td>
<td>instance</td>
<td>J. Smith</td>
</tr>
<tr>
<td>improper noun</td>
<td>class/type/role</td>
<td>toy</td>
</tr>
<tr>
<td>doing verb</td>
<td>operation</td>
<td>buy</td>
</tr>
<tr>
<td>being verb</td>
<td>classification</td>
<td>is an</td>
</tr>
<tr>
<td>having verb</td>
<td>composition</td>
<td>has an</td>
</tr>
<tr>
<td>stative verb</td>
<td>invariance-condition</td>
<td>are owned</td>
</tr>
<tr>
<td>modal verb</td>
<td>data semantics, pre-condition, post-condition or invariance-condition</td>
<td>must be</td>
</tr>
<tr>
<td>adjective</td>
<td>attribute value or class</td>
<td>unsuitable</td>
</tr>
<tr>
<td>adjectival phrase</td>
<td>association operation</td>
<td>the customer with children the customer who bought the kite</td>
</tr>
<tr>
<td>transitive verb</td>
<td>operation</td>
<td>enter</td>
</tr>
<tr>
<td>intransitive verb</td>
<td>exception or event</td>
<td>depend</td>
</tr>
</tbody>
</table>

*http://uml-tutorials.trireme.com/uml_tutorial_4.htm*
Summary

• Our purpose of analyzing the textual description is not a theoretical exercise.
• It enables us to ask questions –
  • Who is doing this action?
  • With what?
  • Why?
  • For whom? Etc.
• It gives us a more structured way to ask questions than simple 5W1H.
• It also constrains us (all structures constrain) and to break that we still need more questions like 5W1H.
Creating a New Visual Modeling Paradigm
Physio-psycho-ethico-sociological parallelism is the inspiration for next few slides.

Prof. R. D. Ranade was one of the foremost teachers of philosophy in India in the first half of 20th century. A master of Greek, Sanskrit, German, Kannada, Marathi and Hindi, he was considered as the last word on Indian Philosophy.
<table>
<thead>
<tr>
<th></th>
<th>Requirements</th>
<th>Design</th>
<th>Code</th>
<th>Test</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Language</strong></td>
<td>Natural Language</td>
<td>UML Informal Diagrams</td>
<td>High level languages</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><strong>Point of View</strong></td>
<td>Customer</td>
<td>Components</td>
<td>Implementation Limitations</td>
<td>End User Behavior</td>
<td>?</td>
</tr>
<tr>
<td>PoV</td>
<td>End user</td>
<td>Deployment</td>
<td>Workarounds</td>
<td>Structure</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Needs and Features</td>
<td>Communication Algorithms/Logic SLAs…</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Model</strong></td>
<td>Free-flow text</td>
<td>Block diagrams</td>
<td>Actual code</td>
<td>Test design techniques</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>Use cases</td>
<td>Flow charts</td>
<td></td>
<td>Tours</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prototypes</td>
<td>Call flow</td>
<td></td>
<td>?</td>
<td></td>
</tr>
<tr>
<td><strong>Challenges/ Sources of issues</strong></td>
<td>Assumptions Understanding Articulation *Transformation…</td>
<td>Assumptions Communication Coupling Logic Transformation…</td>
<td>Assumptions Testability Understanding Transformation…</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Assumptions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Understanding</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Articulation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>*Transformation…</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Person</strong></td>
<td>Customer/B.A.</td>
<td>Architect</td>
<td>Developer</td>
<td>Tester</td>
<td>User</td>
</tr>
<tr>
<td></td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td><strong>Elaboration</strong></td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
<td>TBD</td>
</tr>
<tr>
<td><strong>Analysis</strong></td>
<td>Linguistic</td>
<td>TBD</td>
<td>TBD</td>
<td>Visual Modeling</td>
<td>TBD</td>
</tr>
</tbody>
</table>

* Transformation refers to change of an idea from one form/description to another.
Problems We are Trying to Solve

• Each stage of software development has its own language. Moving from one stage to another requires conversion from one language to another.
• Each model has different Point of View (Pov)
• How do we address
  • “Lost in translation”
  • Impedance mismatch
  • Assumptions
• What are these various languages and models?
• How can I analyze these?
• Can I use models developed at each stage to create my own model?
  • For understanding
  • For unearthing the assumptions and the translation issues
  • Visualize the working of the software
  • Create mental models for use in testing
  • Test idea generation
Combining Models and Analytical Techniques

• Requirements – context-free and contextual questions for natural language, analysis of Use-cases etc.
• Design – Various UML diagrams, ER diagrams, state diagrams
• Code – Call flow
• Testing – Test design techniques, Tours
• Each model fleshes out certain point of view (PoV)

• How can we combine these to create something which is larger than the sum of individual models?
Visual Modeling

What should Visual Modeling be able to do?

1. Be able to reveal the assumptions made as they are a big source of potential defects
2. Be able to reveal the impedance mismatch of “transformations” as these result in defects
3. Be able to utilize various PoV to get better understanding and hence create better models
Visual Models

• Linguistic Analysis + Analysis of Design + Tours = Visual Model

• Questions (Contextual + context free) + Textual/Linguistic analysis + All models + (Functional Testing Heuristics) + Tours + Unified Test Design Technique + Test Design Techniques = Visual Model

• We can combine many more PoVs, models to create our own model while exercising care not to make models which are too time-consuming, complex and not delivering value.
• Models are means to an end, not an end by themselves.
Types of Models – 20,000 Feet View

• View
  • Static
  • Dynamic
• Spatial
  • Paths
  • Deployment
• Temporal
  • Sequence
  • Parallel
  • Synchronicity
  • Time triggered
• Relationship and Hierarchy
• Communication/Flow
  • Business process
  • Workflows
Types of Models – 20,000 Feet View

• Dynamic
  • Event based – State driven or free-form
  • Flow based
  • Use case/Scenario

• Data
  • Logical
  • Physical
  • Database or free-form
  • Transformation

• A more complete list can be derived from the “parallelism” table
Point of View (PoV)

- PoV refers to the focus of the model
  - It is the type of model created at design stage
  - It is the type of the tour you plan to conduct
  - It is the type of model you create while designing tests
Various PoV on the System
Some Details on PoVs

- Consists of
  - Has properties
  - Interacts with
  - Used by
  - Operated by
  - Using means of
  - Has a purpose
  - Has constraints
  - Has features
  - Code/Design

- Operational
  - User
  - Client
  - Admin
  - Development
  - Testing
  - Automation
  - Installation
  - Support

- Users/Role/Purpose
  - Constraints/Rules
    - Max/Min
    - Static/Dynamic
    - Initialization/Configuration data
    - TimeSensitive (Date/time)
    - Timing constraints (real time)
    - ranges/permitted values
    - Default
    - Refreshing/Refresh rate
    - Computations
    - Transformation
    - Comparison
    - Addition/Deletion/Modification
    - Trigger based operations
    - Relationship with other data/datum
  - Operations
    - Atomicity
    - Importance
    - Properties
      - frequency of occurrence
      - Size
      - Programming language representations

- Interface
  - Access control
  - GUI
    - API
    - CLI
    - Protocol
    - File Exchange
    - Webservices
    - H/W-S/W

- Test data
  - Customer data
  - Anonymized data
    - partial/subset
  - Test data types
  - Test data patterns

- Temporal-Spatio
  - Timing of operations
    - Space???
      - Disk
      - Memory
    - Locale
      - System changes with time/Space

- Data
  - System/User
    - Input/Output/Reference (static)
    - Computed/Generated/Inputed
  - User entered/System Generated/Received
  - Event (asynchronous) v/s synchronous
Types of Users

http://testinggarage.blogspot.in/2012/08/context-free-mind-map-of-users.html
Modeling Primitives

Sequential Activities
- Step 1
- Step 2
- Step 3

Activity 1
- Action 1-1
- Action 1-2

Activity 2
- Action 2-1
- Action 2-2

Concurrent Activities

Confluence

Activity Related Primitives
- Action 1
- Action 2
- Action 3

Interaction

Synchronization

Activity Related Primitives
- Feedback

Relationship Related Primitives
- Sharing
- Entity 1
- Entity 2
- Entity 3
Joris Meerts describes a large number of such primitives in his fantastic paper http://www.testingreferences.com/docs/Functional_Testing_Heuristics.pdf

- He describes these as heuristics. I would like to consider them more as different ways in which program artifacts interact or influence.
- I would like to categorize them based on PoV, their dynamic or static nature etc.
- Two such categories were on previous slide
Our Modeling Language

• While the work evolves we want a modeling language which is
  • Simple so that everybody can use it
  • Flexible so that people can customize it to suit their needs
  • Powerful so that it caters to transformations of a model (requirement) into another model (design) to code (which is modeling what the user wants) and so on.
  • Informal
  • With all the elements to convert it into a formal language*

* UML like formal/semi-formal languages are used by people as they want it and not necessarily as the designers intended. Formal languages are harder to use, limiting their usefulness
Creating a new type of informal, visual model (180 minutes)

- Let us start with a simple language for describing the model
- Model we are interested in is
  - Informal
  - Behavior driven
  - Dynamic
  - Ability to include various PoV
- Let us model a small and simple problem
- Let us use this model for designing tests
- Let us extend the model and the language to describe it
- Let us finish with an exercise
Please Check

- Are you able to access
  https://www.softaculous.com/demos/WebCalendar - (admin/pass)
- If not then please look at the next two slides

- Alternate
  - http://www.opensourcecms.com/demo/1/182/Webcalendar - (admin/demo123)
  - http://webcalendar.sourceforge.net/demo/view_r.php?id=557 - (demo/demo)
WebCalendar – Add Entry

Add Entry (event)

Details | Participants | Repeat | Reminders

Brief Description: 

Full Description: 

Access: Public

Priority: 5-Medium

Category: 

Edit
WebCalendar - Categories
Our Model Language

- Simple language for our modeling
- Consists of symbols
- Later you can create your own language
Symbols

Element of interest -

Variability/Width -

Movement –

Reversible/Both ways -

Database –

Permitted Operations -
Model

• We can use screenshots to start creating our model OR hand-draw sketches
• Indicate the properties we are interested in
• Draw more and show relationships
• Show dynamism by using appropriate symbols
Creating a Model

• Next few slides explain creation of a model for addition of a calendar entry. This model is for demonstration and hence “incomplete.”
Analysis

Add Entry (event)

Details

- Brief Description:
- Full Description:

Size, number and type of characters

Source of dropdown entries?

Access: Public
Priority: 5-Medium
Category:

Read-only
Analysis – Category List

[Hand-drawn diagram showing a user interface with options for editing and category management.]
Movement of Elements Between Lists and Reflection
Reversibility, Selection and Other Interesting Things
Placing Things and Moving Around
Thinking Data

DATA
1. WHAT
2. FROM WHERE?
3. TO WHERE?
4. CARDINALITY
Thinking Operations

[Diagram showing operations permitted and related elements]

[Another diagram showing categories and related operations]
Completed Model

DATA
1. WHAT
2. FROM WHERE?
3. TO WHERE?
4. CARDINALITY

EDIT

<< CATEGORIES >>

OPERATIONS PERMITTED?

MECHANISM OF TRANSFER?
(URI/POOL/HTTP)

DIFFERENT FROM STANDARD GUI?

GUI
RESIZE?

OK
CANCEL

INCONSISTENT?

EFFECT?
Connection Tours, Testing and Models

• Tours are PoV
• Can be used to explore applications
• Can be used for testing
• One way to look at models is that we put our understanding on paper
• Tours help us develop that understanding and hence models
• Type of model $\leftrightarrow$ Type of tours $\leftrightarrow$ Described model
Some Good Modeling Practices

• Identify the key PoV/Perspectives
• Choose a PoV and try to complete it
• Change your PoV when you get stuck. Focus-defocus heuristic comes to mind (James Bach’s heuristic)
• Evil Tester’s heuristics
  • Big picture – chunk up and down
  • Remodel – list/time/attribute/specs/flows etc.
• There can be different types of models
• No model is ever going to be complete nor should that be the intention
• Whenever you get new ideas revisit your model and make changes
• Whenever you actually test the software, the model will necessarily need to be changed because dynamism brings in a whole lot of new information that will require you to revisit your assumptions
Some Food for Thought - Visual Test Model

• Capturing, planning and presenting Test Coverage using diagrams
• Mind maps, flow / basic architecture diagrams
• Map functions within the system using code structure
  • a search function found on 3 different screens may be using the same underlying code.
  • search would appear on the system map once, with a reference to the various screens which call it

• Purpose
  • To clarify scope and priority
  • Understand potential test coverage
  • An alternative to a test script
  • Impact analysis
  • Work allocation

• Leah Stockley - http://www.inspiredtester.com/1/post/2013/04/visual-test-models.html
Creating a visual modeling toolbox

- Software Ideas Modeler
  http://www.softwareideas.net
- Online tool for BPMN, Flow charts and others -
- Visio
- UML diagramming tools
- Whiteboard
- Pencil and paper
- Mind mapping software
- Tablets

Evil tester suggested tools*
- Graphviz
- yED
- Freelance
- Evernote
- yUML
- Umlet
- Asciiflow
- Drawtiming
- Paint.net

*https://www.eurostarconferences.com/media/149386/alan-richardson_virtual-conference.pdf
Potential Issues in Using Models

• Models can never be complete and in many cases accurate
• Models make concrete the creator’s understanding and you might mistake that concreteness with the real product binding you to the model
• Inspirations, after a while, are difficult to get from the models
• Limitations of model or the representation of the model can make you stagnate and you may miss things
• ....
Thanks

Questions?
Appendix
Exercise

• Spend 30-45 minutes in groups of 3-4 persons each.
  • Half the groups will do User Tour of the application and create enhanced model of what we did.
  • Other half of the groups will do Feature Tour of Import-Export functionality of events and create a model connecting with the model we created.
  • Both groups will look at Delete Entries under event and Today + This Week under My Calendar.
  • Feel free to copy the images from http://shishya-eternalstudent.blogspot.in/ for the model already done.
  • If required create non-admin users and other types of users.

• Additional points
  • Do you want to create some more symbols to make your model more clear/complete? Which ones?
  • Would you want to use some modeling primitives from Joris Meert’s Functional Testing Heuristics paper?
Some Questions

• Which of the UML and other models you would like to use?
• Would you like to use Linguistic Analysis or Unified Design method? How?
• How and where would you want to use
  • mind maps?
  • context-free questions?