Usability Analysis of WB-Graph tools

June 6, 2005

Presentation of the Diploma Thesis ’Formal Task Analysis of Graphical System Engineering Software Use’

What is Usability?

- often described as the "ease of use" (Macauley1995)
- more complete
  - ease of learning
  - ease of use
  - flexibility of use
  - effectiveness of use
  - user satisfaction with the system

The User Interface

- not just the Graphical User Interface (GUI)
- how tasks are accomplished with a machine
- what you do and how it responds (Raskin2000)

Direct Manipulation

- the user manipulates graphic representations of underlying data
- Principles (Shneiderman1982):
  - continuous representation of the object of interest
  - physical actions or labelled button presses instead of complex syntax
  - rapid incremental reversible operations whose impact on the object of interest is immediately visible

Engagement of the User

- direct engagement
  - user feels as direct actor
- indirect engagement
  - hidden intermediary who executes commands

YB-Edit

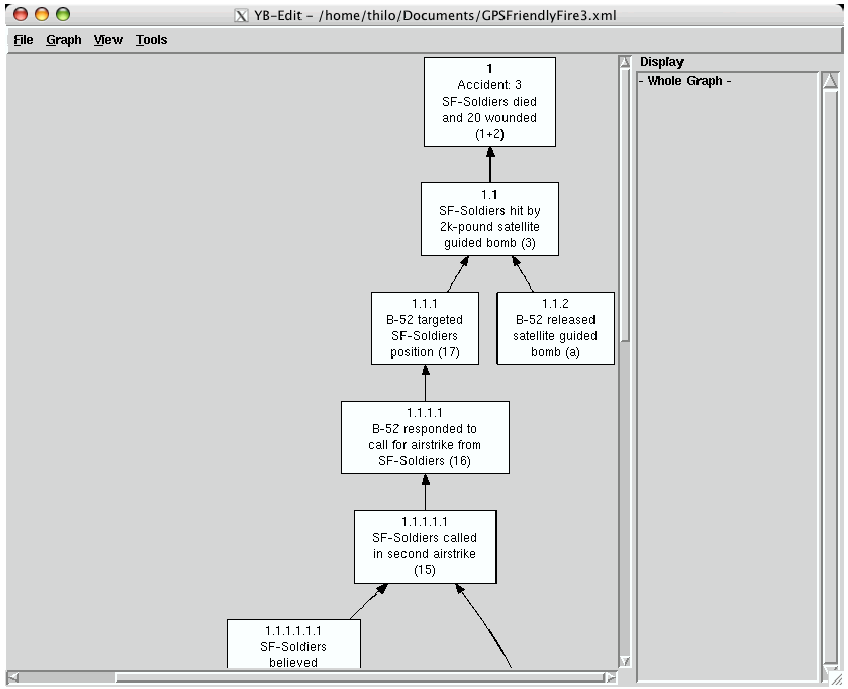

Figure 1: Program window.

Figure 2: Context pop-up menus: node menu, edge menu, and canvas menu.

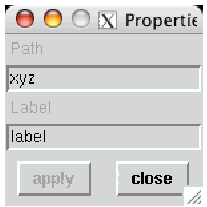

Figure 3: Pop-up input dialogue window.

Analysis Methods

Hierarchical Task Analysis (HTA)

HTA

- introduced by John Annett and Keith Duncan for training applications in 1967
- general method for task representation
- the level of detail arises from the motive of the analysis and the kind of the analysed task

HTA

- produces a hierarchy of
  - operations
  - plans
  - which must be carried out to get a task done
- examination of
  - operator’s resources, constraints, and preferences
  - and their influence on human operations in attaining the system goal

Procedure of a HTA

- state the goal
- describe the goal as set of sub-operations leading to the goal and plans when to carry them out
- further description of sub-operations as set of sub-operations and plans when necessary or wanted

Goals, Operators, Methods, and Selectors Language (GOMSL)

GOMSL

- introduced by David E. Kieras for user interface design in 1995
- formalised version of Natural Goals, Operators, Methods, and Selectors Language (NGOMSL)
  - introduced by David E. Kieras for user interface design in 1988
  - structured natural-language notation for GOMS models with advanced support for cognition

GOMSL

- based on Goals, Operators, Methods, and Selectors (GOMS) introduced by Stuart K. Card, Thomas P. Moran, and Allen Newell for analysis of text processor use in 1983

- method to produce quantitative and qualitative predictions of single user interaction with passive and active systems

GOMSL

- serial stage architecture analysis with program like structures
- the execution times of the operations needed to achieve a goal is added up to make time predictions
- can be interpreted with GLEAN3, a software tool which does all the calculations of the analysis

The Analyses

HTA of WBA

- purpose:
  - determine the high level goals of YB-Edit use
  - gain an overview of the WBA method
  - data acquisition through interviews

Performing a WBA

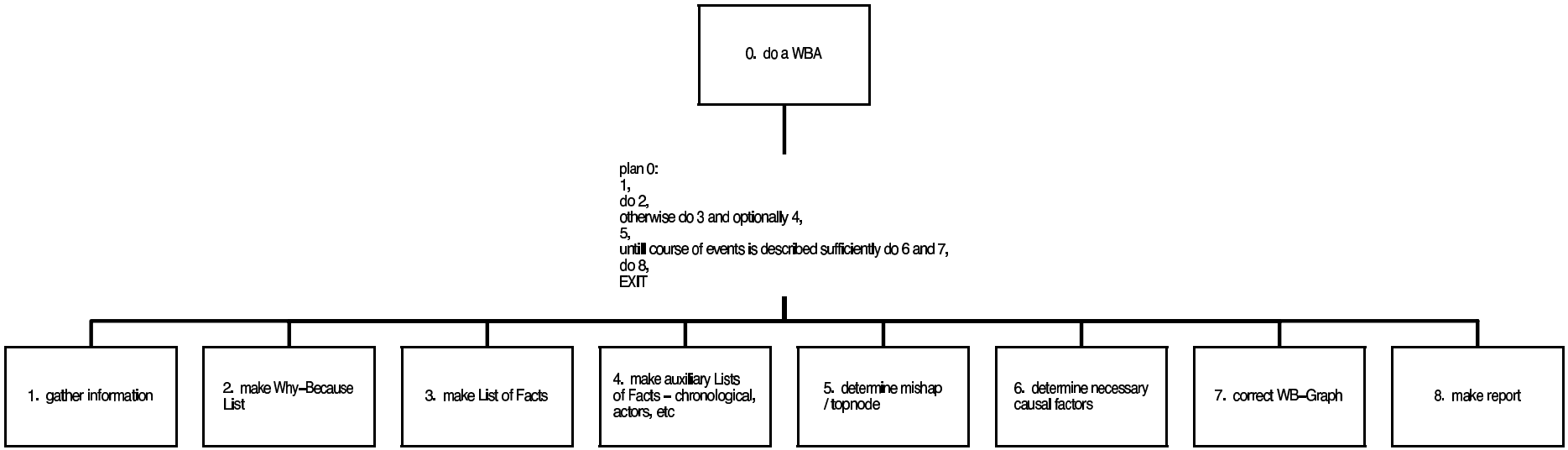

Figure 4: Performing a WBA.

Correct the WB Graph

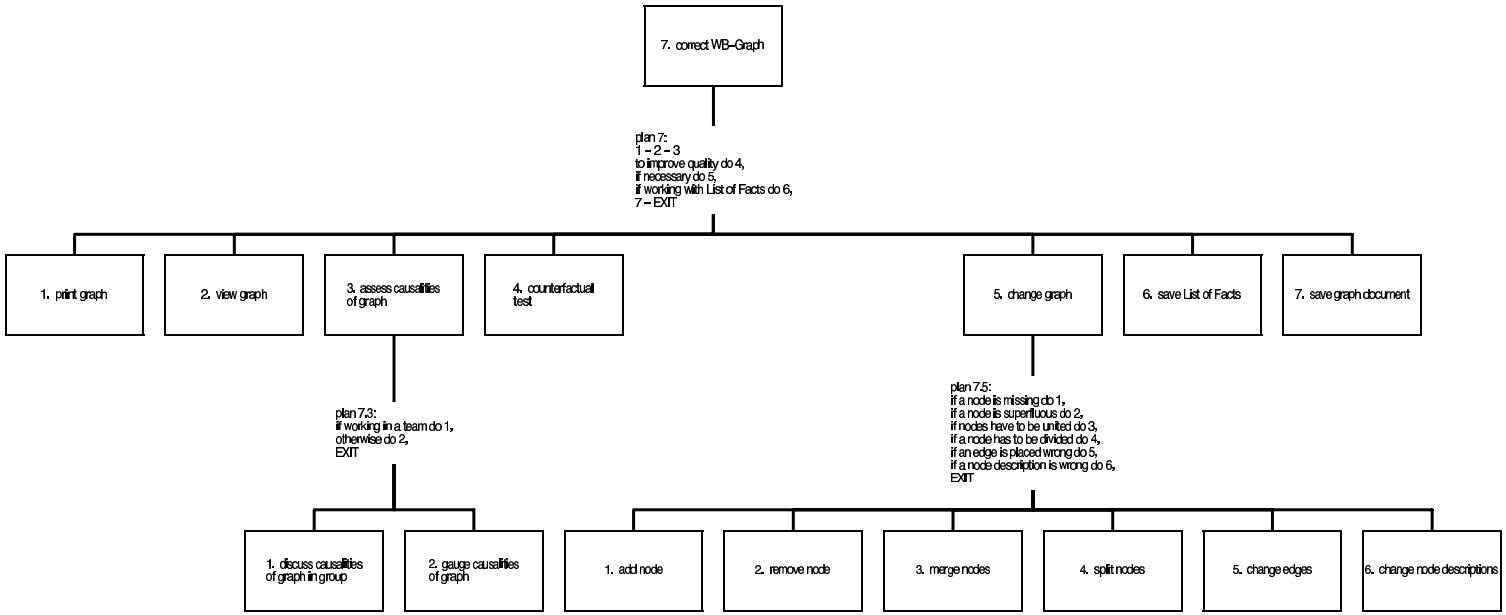

Figure 5: Correction of the WB Graph

Adding a Necessary Causal Factor or Node

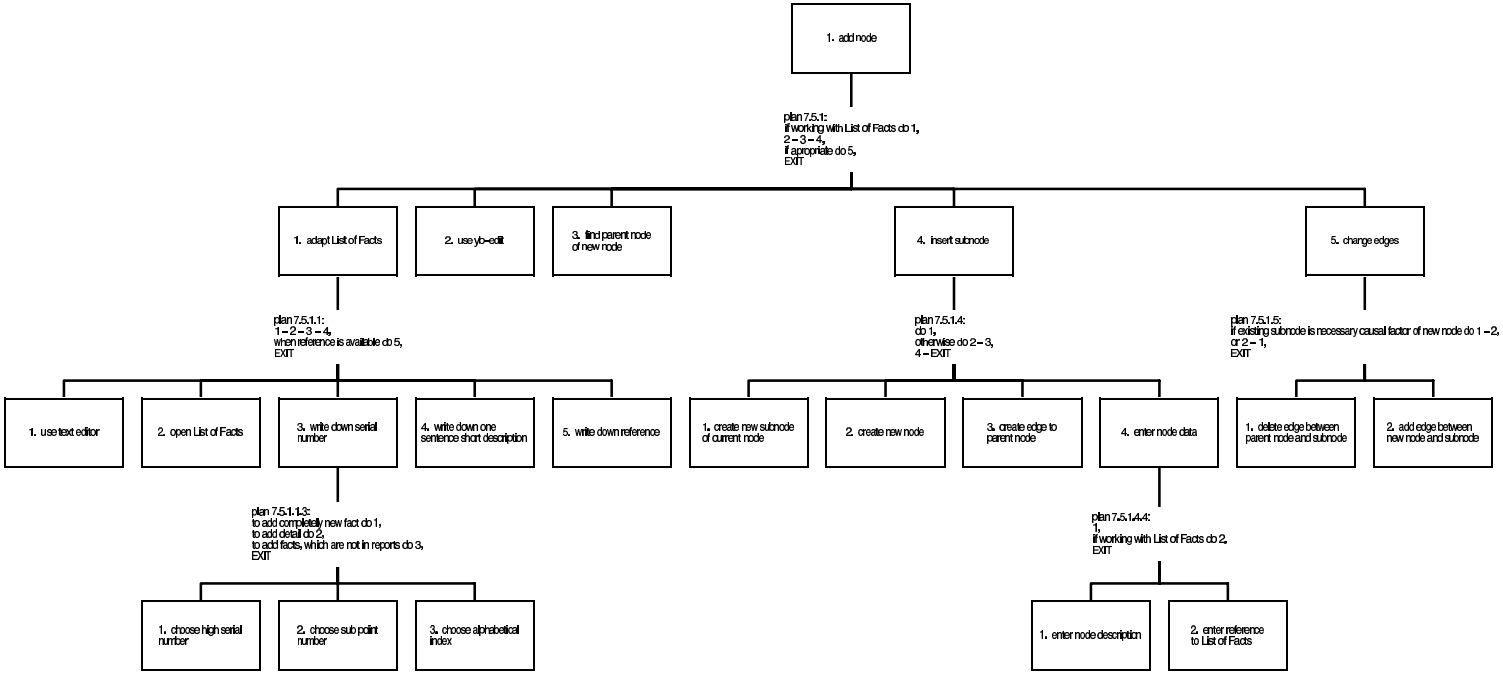

Figure 6: Adding a node to the graph.

Results

- high level goals of YB-Edit use were determined by analysing the task and not by YB-Edits interface
- complete process of doing a WBA is described
- can be extended (insertion of subgraphs, e.g.)

GOMSL Analysis of YB-Edit use

- purpose:
  - determine how the previously defined high level goals are accomplished using YB-Edit
  - detect possible improvement opportunities or sources of error
- case analysed: ’Friendly Fire’ Deaths Traced to Dead Battery: Taliban Targeted but U.S. Forces Killed
- GOMSL model represents creation of ’Friendly Fire’ graph using YB-Edit

Device Description

- the GUI
- Device Description: GUI, Program window of YB-Edit

```
Visual_object: Ybedit_program_window
  Type is window.
  Label is "YB–Edit".
```

- Device Description: GUI, Menu entry

```
Visual_object: Add_new_node
  Type is menu_entry.
  Label is "add␣new␣node".
```

Device Description

- Device Description: Node in WB-Graph

```
Visual_object: Node_11
  Type is node.
  Label is "SF–Soldiers hit by 2k–pound satellite guided bomb".
  Reference is "(3)".
  Children is "2".
  Child1 is "B–52 targeted SF–Soldiers position".
  Child2 is "B–52 released satellite guided bomb".
  Parents is "1".
  Parent1 is "Accident: 3 SF–Soldiers died and 20 wounded".
```

Device Description

- Device Description: Edge in WB-Graph

```
Visual_object: Edge_11–1
  Type is edge.
  Source is "SF–Soldiers hit by 2k–pound satellite guided bomb".
  Target is "Accident: 3 SF–Soldiers died and 20 wounded".
```

Task Description

- a sequence of tasks consisting of the previously defined high level goals
- Task Description: Task for adding a new node to WB-Graph

```
Task_item: Task4
  Name is Task4.
  Type is add_ncf.
  Label is "SF–Soldiers␣hit␣by␣2k–pound␣satellite␣guided␣bomb".
  Reference is "(3)".
  Parent is "Accident:␣3␣SF–Soldiers␣died␣and␣20␣wounded".
  Next is Task5.
```

Methods and Rules Description

- how the high level goals are accomplished using YB-Edit
- Method Description: Adding a no to the WB-Graph

```
Method_for_goal: Add ncf using
                 <ncf_parent>, and <ncf_label>,
                 and <ncf_reference>, and <ncf_node_number>
  Step 1. Accomplish_goal: Issue context_command using
                           "add ncf", and node,
                           and <ncf_parent>, and nil.
  Step 2. Accomplish_goal: Enter ncf_description using
                           <ncf_label>, and <ncf_reference>,
                           and <ncf_node_number>.
  Step 3. Return_with_goal_accomplished.
```

Results

- all high level goals can be reached using YB-Edit
- model consists of 34 different methods with a total of 216 steps, 2.78% are learnable
- building the ’Friendly Fire" graph took 704.95 sec

Results

- a lot of time (40%) was spent entering text
- changing forth and back to an editor to keep the List of Facts up to date consumed 16.6% of the time

Results

- two basic interaction forms
  - pop-up menus (indirect engagement)
  - drag & drop (direct engagement & manipulation)

Results

- input dialog window is inconsistent in invocation and use
  - four different ways to let the window pop up, which control the state of the window
  - window has four different modes without clear signs to detect

Results

- external program to manage the List of Facts
  - requires a lot of user interaction
  - is time consuming
  - errors due to
    - problem of closure
    - manual text entering

Results

- insertion and removal of a node require a lot of manual operation
- selection is not possible, therefore only single object manipulation

New User Interface Design

Concepts

- interaction through on gesture with direct engagement & manipulation
- consistent text input
- incorporation of the List of Facts
- automation:
  - insertion of node(s)
  - removal of node(s)
  - indexing the graph

GUI Sketch

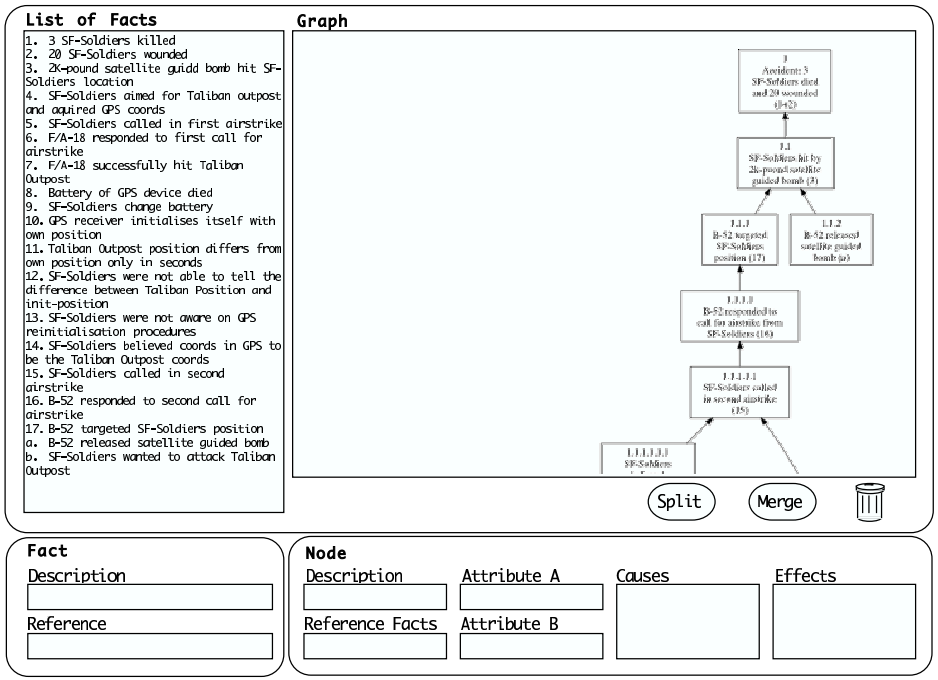

Figure 7: Sketch of a new GUI design

Results

- model consists of 22 methods with a total of 140 steps, 5% are learnable
- building the ’Friendly Fire" graph now took 236.55 sec
- problem of closure does not occur any more

Conclusion

- combination of HTA and GOMSL/GLEAN3 was very useful to examine the use of YB-Edit in context of WBA
- experience helps and influences the quality of the models

GLEAN3 deficiencies:

- no loops can be specified
- size and position of visual objects can not be specified
- no way to control the visibility of visual objects
- only global pseudo-parameters ("variables")

YB-Edit use

- analysis showed several areas of possible optimisation
- new interface design
  - was directly based on the high level goals
  - speeds the interaction up by factor of three
  - disburdens the user in several areas
  - avoids two possible causes for problems of closure

Future Plans

Cockpit Use Analysis

- transfer concept of GOMSL analysis to analysis of cockpit use in hard real time environments (car, train, ...)
- simulate cockpit use with respect to the specified environment with a tool like GLEAN

Cockpit Use Analysis

- specify in a formal, easy to learn language:
  - the environment
  - the interface and methods to use it
  - the tasks to accomplish

Benefits

- analysis can be made at very early design stages
- specification of models in relatively short time (magnitude: man/month)
- models are reusable

Possible Application

- analyse interaction with distributed system components (car control, communication systems, navigation system, ...)

- detect
  - if demands of hard real time environment collide with cockpit interaction design
  - if demands of various components collide with each other
  - possible causes of user error

A Tool: script2graph

script2graph

- tool to convert CI- and WB-Scripts to dot, vcg, or aisee format
- supports grouping of nodes
- uses RecDescent parser module
- readable perl code, easy to customise

Example:

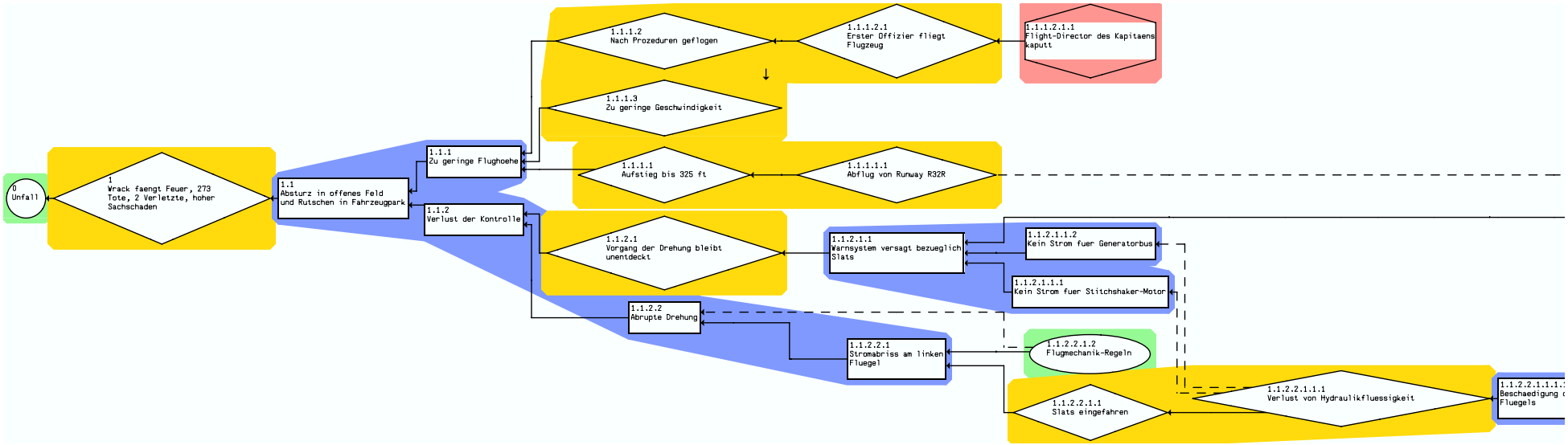

Figure 8: Coloured regions

Thank You!