

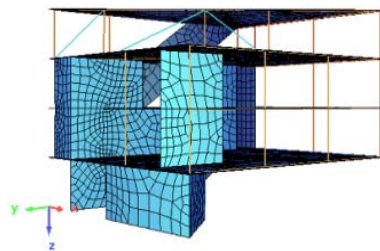
Tragwerk-FMEA

D8 Software Description

Tesfaye Regassa, Peter Struss
MQM Group, Technische Universität München

Michael Eisfeld
Eisfeld Ingenieure

20.11.2010
Version: 1.2



Contents

Tragwerk-FMEA	1
D8 Software Description	1
0 Document History	2
1 Introduction	2
2 Application Environment	2
3 Software	3
4 Functional Requirements.....	3
5 General Solution Idea, Architecture.....	8
5.1 Solution Ideas	8
5.2 Architecture, Concepts and Interfaces	10
5.3 Interface Design	13
6 Data Structures	18
6.1 Conceptual Classes.....	18
6.2 Data Exchange with Kassel Modules.....	18
6.3 Data Exchange with QFE:.....	20
6.4 Data Exchange with TableEditor	21
7 A Session with the Structure-FMEA Tool	22
7.1 Starting StructureFMEA	22
7.2 Opening the Structural Concept	23
7.3 Selecting Critical Elements	23

7.4	Modes of Operation.....	23
	Interactive Mode of Operation	24
7.5	Editing the Results Table	26
8	Benefits of Using the Tool	27

0 Document History

Version	Date	Changes	Author
1.0	19.11.2010	1 st draft	Struss
1.1	19.11.2010	Modified figures 4, 6 and 8, revise section 6, added Section 8	Regassa
1.2	20.11.2010	Introduction, minor revisions	Struss

1 Introduction

This document describes the software tool that implements the automated FMEA of structures according to the guidelines outlined in deliverable D3 and based on the conceptualization described in deliverable D6.

It lists the requirements, describes the principal ideas underlying the solution and the architecture of the software, the interfaces between the different modules and the data structures. It illustrates a session with the tool through screenshots and discusses the benefits of using the tool compared to a manual process.

2 Application Environment

The software shall be deployed as a desktop application. It shall enable design engineers and planners of a civil engineering structure to assess the potential risk on the overall stability of the structure due to failure or weakening of critical structural elements. The software would be put into operation in conjunction with a Statics/FEM application to utilize the computational capabilities of such software for necessary numerical iterations and a structural BIM-application that provides information about the global load transfer, the element capacities with its thresholds, the original internal forces the structure was designed for and the part-of hierarchy. In addition to the FMEA tool, there is a module that generates the influence functions from the FE-results and the structure of the BIM-software, a 3D tool to

display the structure and the results of the FMEA and an algorithm that steers the global progressive collapse by constantly calling the FMEA-module until the progression stops.

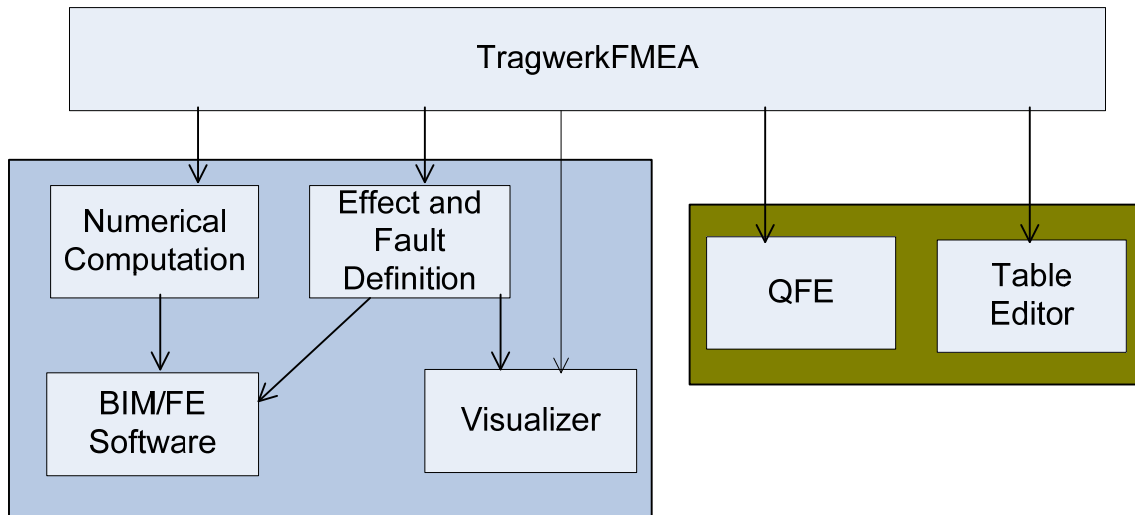


Figure 1 Application environment

3 Software

List of external (3rd party) software necessary for implementation of the system

- BIM software
- 3D Visualizer
- Statics/FE application
- QFE

4 Functional Requirements

/LF10/ Select Structure

Actor: Tester

Description: To start analysis, the user/Tester should be able to select a statically isolated sub-structure that (s)he wants to examine. This loads structural data from a data file or imports it from an external CAD/Statics/FE application into the Project Viewer. This requires an interface to the external applications. The user imports the so called structural concept that includes the system with its structural elements (internal forces, element capacities, geometric information from which the element stiffness matrix can be derived and the element's connectivity to other members), the loads acting on it and the global load-transfer graph.

/LF20/ Select critical elements

Actor: Tester

Description: From the 3D display of the structural concept, the user selects parts (single elements or connections) of the structure that he presumes to be critical (**CriticalElement**). This concerns elements whose failing is considered important (**CriticalFaultElements**) and elements for which the impact of a failure is relevant (**CriticalEffectElements**). The selection can be based on experience of the user, on preliminary analysis or on preset risk priorities.

/LF30/ Derive influence functions

Actor: System

Description: this derives influence functions as the effect of possible failure or weakening of critical elements (**Fault** of a **CriticalFaultElement**) on selected elements for which the impact of the fault is considered relevant (**CriticalEffectElements**). This will be a sort of a cause-and-effect matrix. The influence functions shall be derived for various causes (Faults or what ifs), e.g. reduction of stiffness (due to environmental effects, accident, prior overestimation, etc.), formation of plastic hinges, collapse, etc.¹.

/LF32/ Generate Thresholds

Actor: System

Description: For the selected **CriticalElements**, depending on the type of each element, the system derives **MemberReaction** (i.e carrying capacity) thresholds. These are (maximum) load carrying capacities of **CriticalElements** (separately or jointly evaluated for axial force, moment and shear).

/LF34/ Define Effects

Actor: System

Description: Based on the computed thresholds and on material properties, the system defines **Effects** for each of the **CriticalElements**. This takes the form of qualitative values, i.e. intervals indicative of the range to which the element is stressed (e.g. elastic range, yielding or collapse). This also includes definition of the possible faults related to the effects.

¹ The input for the matrix computation is derived from the structural concept using ConEd

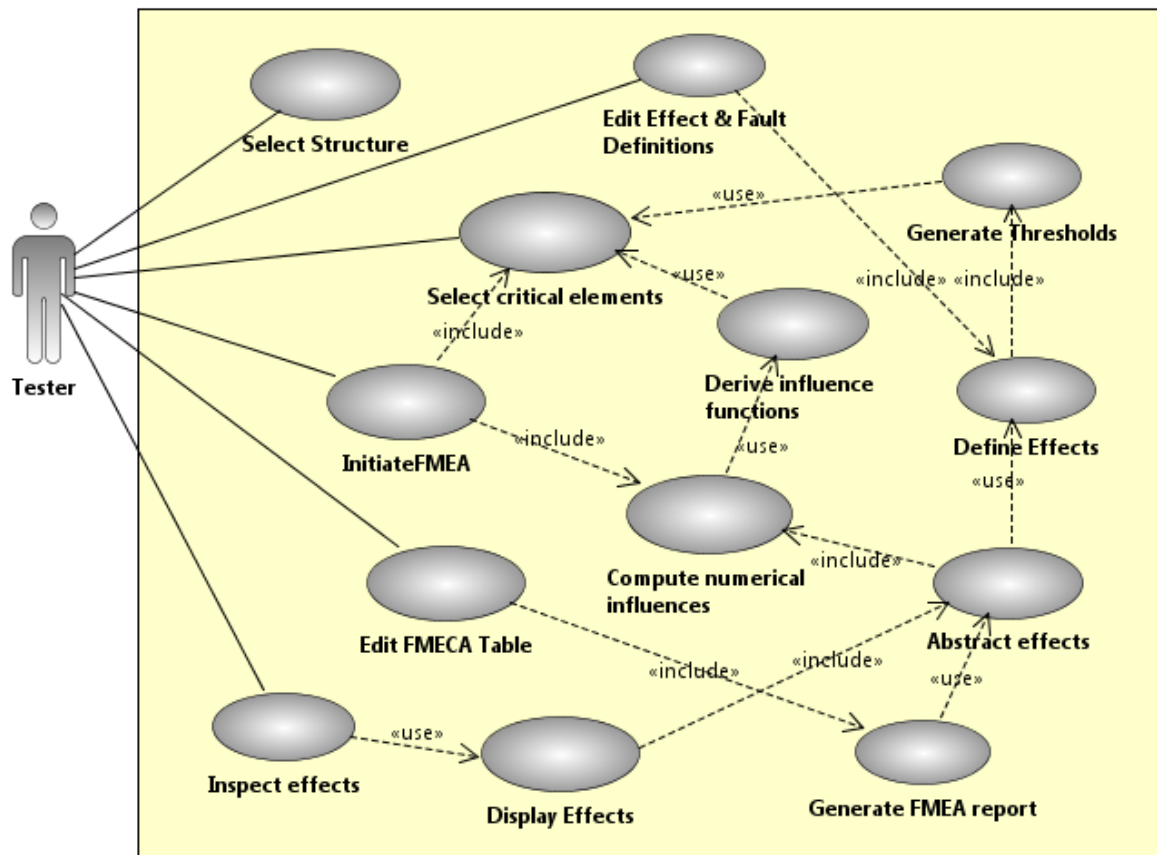


Figure 2 Top level use cases

/LF36/ Edit Effect & Fault Definitions

Actor: Tester

Description: The Tester should be able to inspect, edit and save the computed thresholds and the Effect and Fault Definitions.

/LF40/ Initiate FMEA

Actor: Tester

Description: To the selected an initial **CriticalFaultElement**, the user assigns a failure scenario (Fault) (s)he wants to check and starts the iteration. The system then should compute the redistribution of the loads in the structure as a result of the fault, using the influence functions. To compute the redistribution of the forces and decide whether or not the structure is stable, a step by step procedure must be applied as described in the progressive collapse algorithm (see appendix). The user must have the opportunity to steer the process if required.

/LF45/ Compute numerical influences

Actor: System

Description: Based on the selected Fault (and subsequently on set of faults that may follow thereof) and using the influence functions, the system computes the **changes** in the MemberReactions in the CriticalEffectElements.

/LF50/ Abstract effects

Actor: System

Description: From the computed numerical influences (redistribution of loads), the software computes the changes in member reactions, and abstracts the qualitative effects on the selected CriticalEffectElements, i.e. it checks whether critical effects have resulted. The system then displays the result of the abstraction. If, for any of the elements under consideration, the resultant effects are such that certain thresholds are exceeded, the system should modify the structural concept accordingly and the process will be repeated with the fault of these new elements as input. The cycle of iteration can be automated or can be steered by the Tester.

/LF55/ Display Effects

Actor: System

Description: The system shall produce a color-coded display for visual inspection of the effects (one time or cascade of effects) caused by Fault of a particular Element. This display shows the critical elements that exist in the load-transfer graph and/or a 3D model. The coloring is done according to the severity of the effect, i.e. based on the “qualitative ratio” of existing forces/element capacities and the saved elastic energy in the structural elements under consideration, according to some thresholds that are pre-defined for each CriticalEffectElement.

/LF60/ Inspect effects

Actor: Tester

Description: The user must be able to visually inspect the result of the effect abstractions and decide whether to continue or break the iteration.

/LF70/ Generate FMEA report

Actor: System

Description: The system generates, as an output, the matrix of cause and effect abstractions of an analysis.

The table should be organized in a way that it displays the interrelationship of elements, faults, and effects occurring at different levels and iteration steps, containing columns for

- StructuralElementId (and/or name)
- Fault (the initial fault of the analysis)
- Local effect (at the StructuralElement subject to the fault; this is a pre-defined text in the description of the fault)
- Propagated effects (of other StructuralElements at the basic level) with indication of the fault cascade (i.e. the iteration step which generated the effect), e.g. by decimal classification
- Subsystem and system level effects, i.e. instability
- The numbers required for criticality analysis
- Consequences etc.

The system includes editing facilities with which the tester can generate reports on risk potentials for the selected critical elements or for the whole structure.

/LF80/ Edit FMECA Table

Actor: Tester

Description: The user must be able to add input to the criticality assessment and to edit entries in the auto-generated FMEA Table with the following functionality:

- Rephrasing fields in the table whose content has been generated automatically (e.g. the wording for the effects)
- Deleting rows in the table in case a generated effect is considered implausible or irrelevant. (If there are remaining effects of a failure, the respective information has to be maintained).
- Adding lines with additional effects caused by a particular failure)
- Deleting all entries for a particular failure (e.g. in case it is considered irrelevant)

- Adding rows for another failure that has not been subject to the automated analysis.
- Persistence of the changes: The editing facility must include ways of saving changed entries between cycles of the FMEA iterations. The system must then request user input before overwriting the changed entries. In detail:
 - Rephrasing of entries should be done for those that are also present in the next run
 - Upon request of the user, deleted and added rows should be displayed and marked, allowing the user to inspect and (dis-)confirm whether changes carry over (all of them or individual ones)

5 General Solution Idea, Architecture

5.1 Solution Ideas

- 1 There exists a focus on both **critical elements** whose faults have to be analyzed as initial faults and on elements for which the impact of a fault has to be determined. Since the impact can be catastrophic, a resulting fault has to be specified. A fault of an element is defined by associating a certain value to the stiffness (longitudinal, shear or bending) of an element (potentially at a certain location).
- 2 The impact of faults has to be evaluated using **numerical computation** which, ultimately, is based on a FE model of the structure. It is assumed that this **computation is captured by influence functions** that compute the impact on a particular member reaction (force, bending moment) at a particular location of a particular element (in the focus).
- 3 The **impact** has to be assessed in a **qualitative** way by a finite number of categorization, such as “no significant impact”, “significant, but not catastrophic impact”, “catastrophic impact”, where the latter corresponds to a failure of the respective element and, hence, results in a modification of the structure if further impact has to be computed. From the FMEA perspective, the first impact category means “no-effect”, whereas the others correspond to effects.
- 4 We assume that this categorization of the impact (on a particular member reaction at a particular location of a particular element) and, hence, the effects can be specified by **thresholds for the respective member reactions** that are given for each (location of an) element in isolation (dependent on the type of element, its parameters, material

etc.). Usually this is defined by relations that have to be valid such as the compression force must be smaller or equal to the compression resistance of the element, or the element must not be under tension. An element may have a few of such relations that must hold. For each critical element, it is necessary to know what the element can withstand - this should be pre-computed and known to the FMEA-module. Initial internal forces are those on which design of the element has been based. Finding the percentage change in exerted forces/moments taking into account the element specific capacity is the major task.

- 5 In general, an effect could be a relation in the Cartesian product of several member reactions. For the prototype to be developed in the project, we assume that effects are specified by **inequalities of one member reaction only**. (Combined loading - a case for columns, frames and trusses - requires complex thresholds, i.e. specified as a constraint over several quantities, which may enforce the introduction of multiple thresholds.)
- 6 On this basis, the **influences** can be **abstracted** to a qualitative level specified by the relevant local thresholds. This is done using the qualitative abstraction operator developed at TUM. If this property is violated, further analysis (computation of extreme points) is required.
- 7 The computation of the effects can then be performed by the **Qualitative FMEA Engine (QFE)** developed by OCC'M SW in the AUTAS project.
- 8 The **result** of this computation is a **list of effects** (if any), i.e. a list of element.location.memberReaction items with an effect category as given above. This output is the basis for two further steps.
- 9 The first one is **visualization** of the effects, e.g. by coloring the elements that suffer from an effect.
- 10 Second, in case of catastrophic effects, the **FE model** has to be **modified** accordingly, if further analysis is requested. We assume that this can be performed automatically (but triggered by a user request). This process stops if parts of the structure become unstable. ("Unstable" refers to the whole structure or a major structural system, whereas 'catastrophic' effect is related to the impact on (collapse of) a single element. Unstable means that there exists a degree of freedom (translation or rotation), whereas a collapse of one element might not lead to instability of the structure since other elements may take up the load as replacement.)

11 In the end, an FMECA table is generated containing the list of fault-effect associations for effects local to elements and for system level effects. This table can be edited by the user. This includes entering required numbers for the computation of the risk priority.

5.2 Architecture, Concepts and Interfaces

The conceptual solution results in an architecture sketched in Figure 3

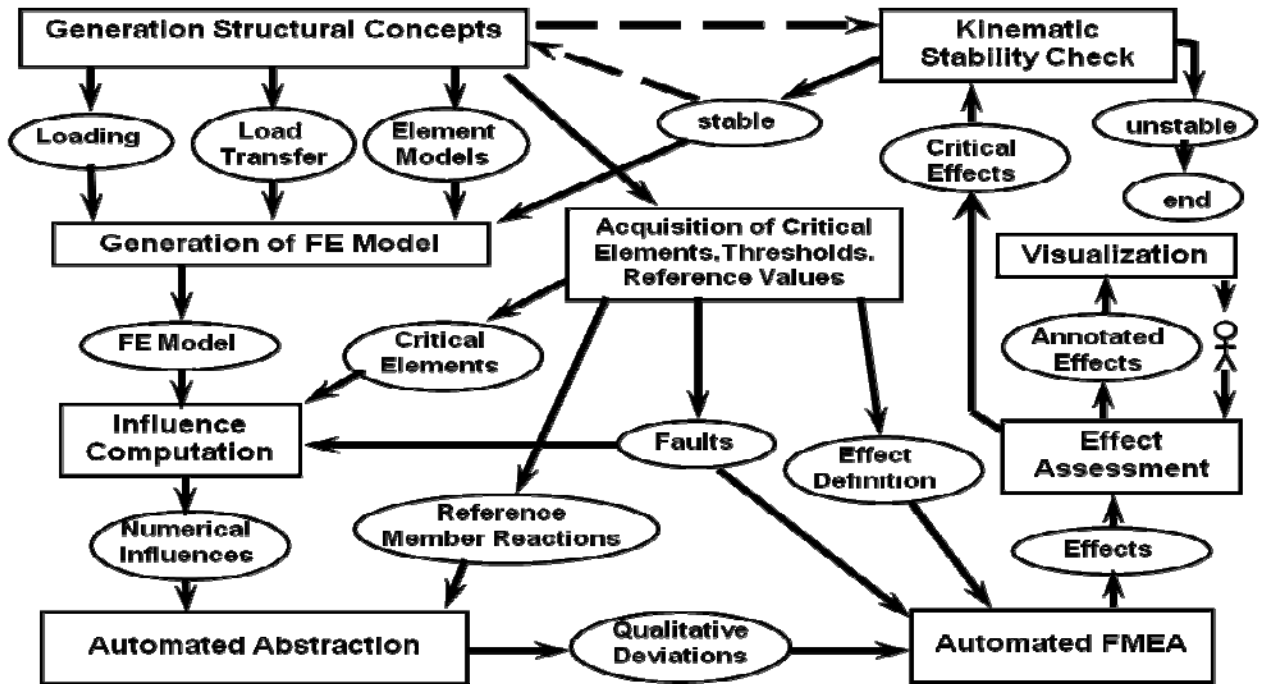


Figure 3 Architecture of Structure FMEA

Figure 5 presents the preliminary component modules

Figure 6 shows the overview of the FMEA activity.

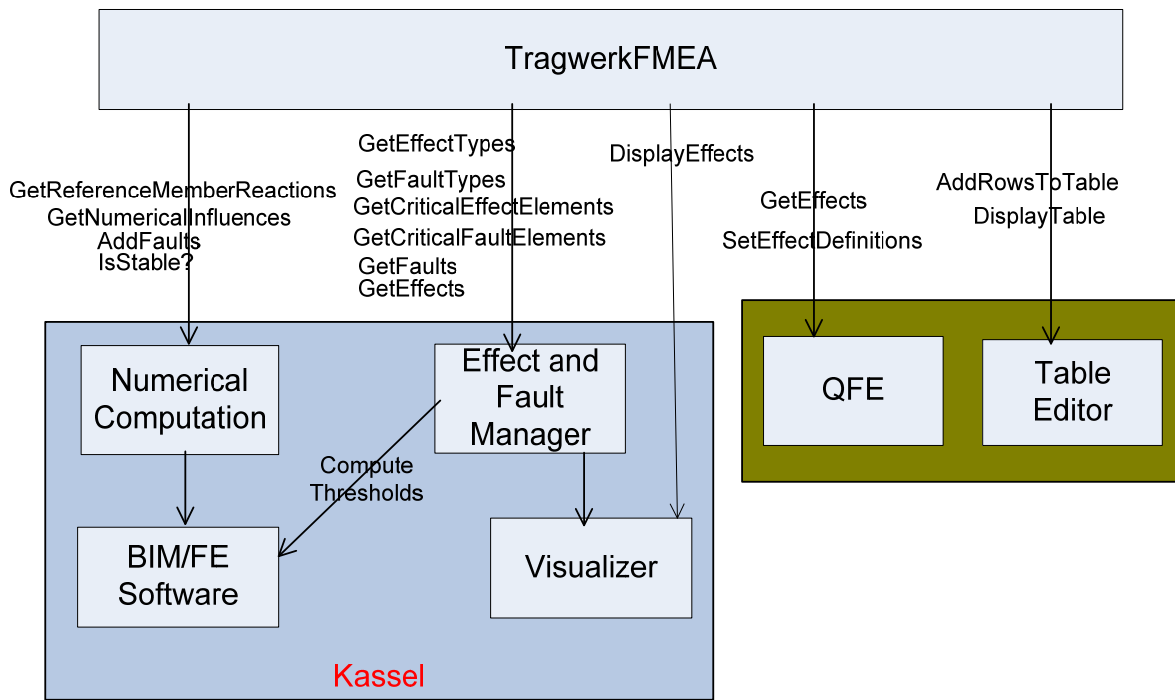


Figure 4: Communication diagram between the modules

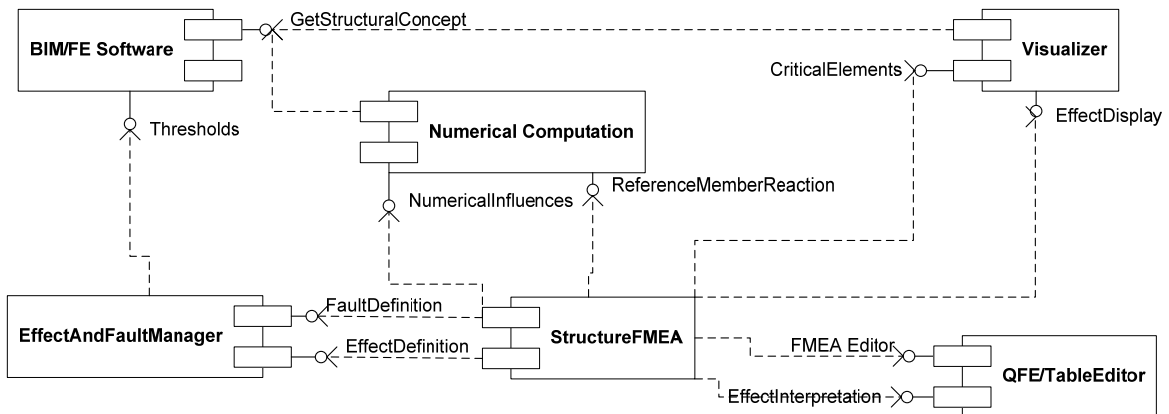


Figure 5: Components Diagram

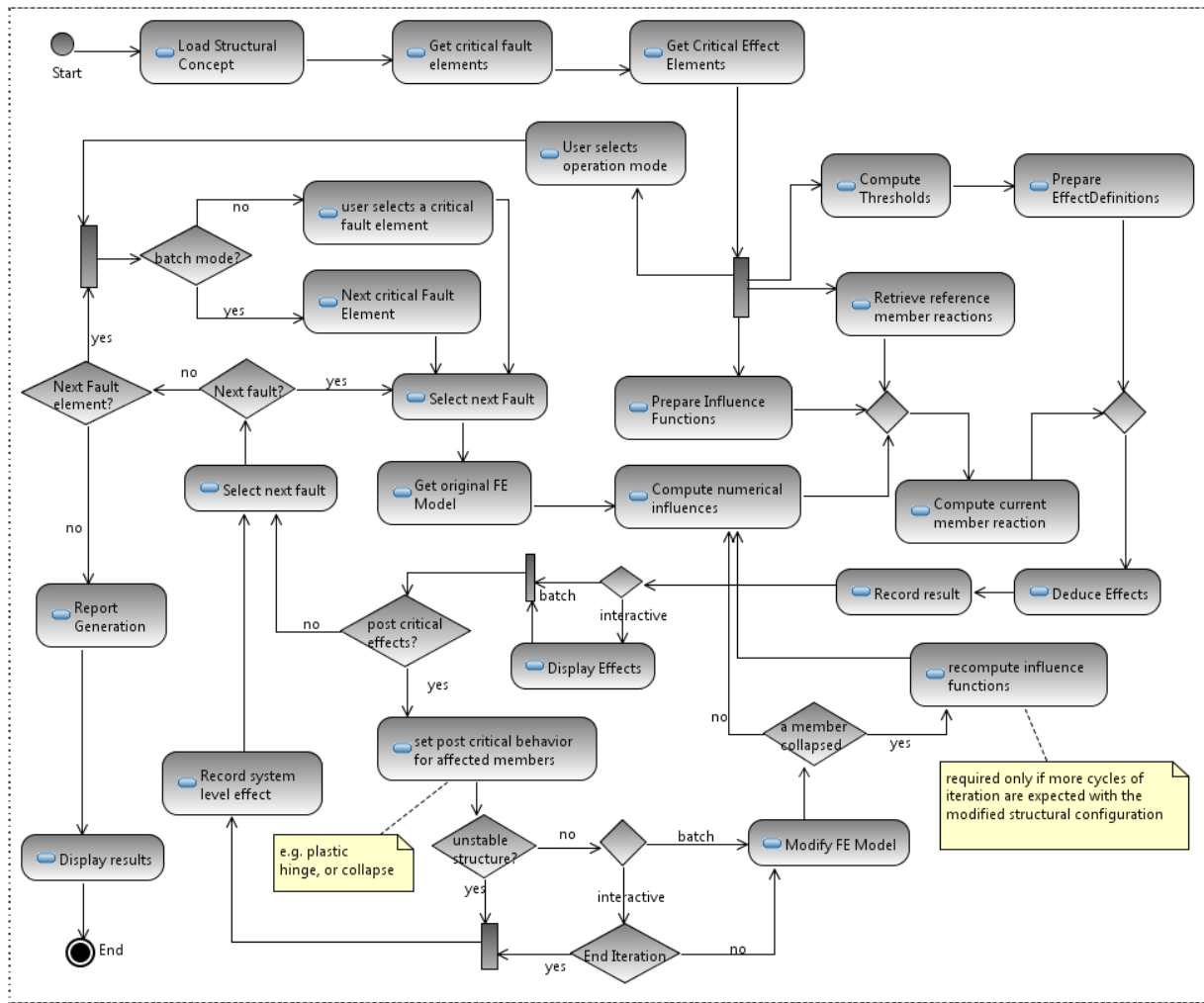


Figure 6 Overview Activity diagram

Figure 7 shows a detailed activity diagram with object flows.

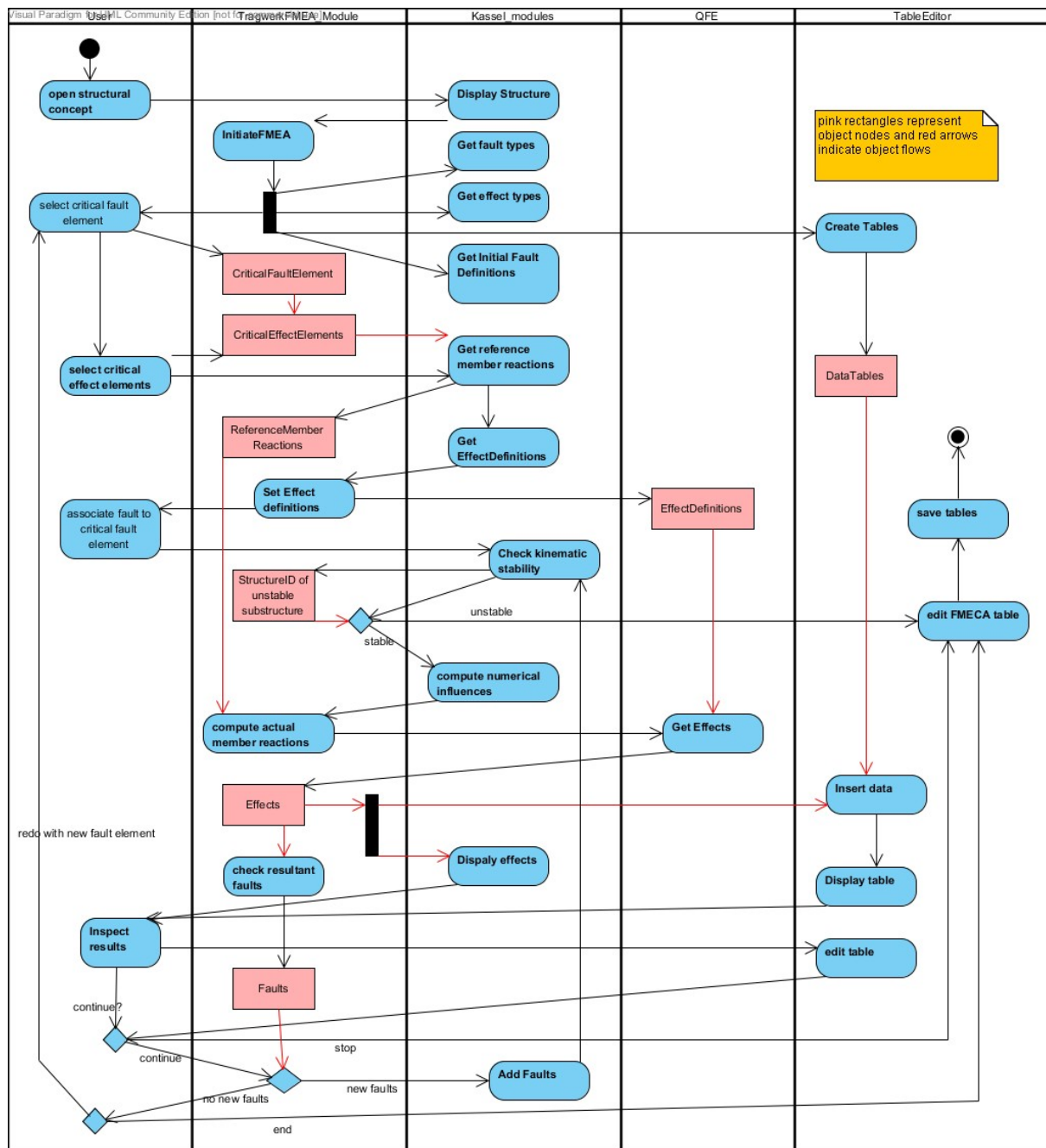


Figure 7 Activity diagram showing object flows during interactive mode of FMEA operation

5.3 Interface Design

As the preliminary component diagram (Figure 5) shows, the module StructureFMEA is a central unit that holds the commands to the other modules. Its basic function is, thus, providing function calls to the rest of the system.

The following listing shows in their order of execution the different methods that should be made available by the `StructureFmea` module.

Module: StructureFmea

1. **OpenStructuralConcept()**: Given a filename as a *string*, **OpenStructuralConcept()** starts the FMEA process by opening a structure concept in the **Visualizer**. It is presumed that the concept is saved as a file. The **Visualizer** then shows a 3D display of the structure model.

```
Visualizer.InitViewer(filename);
```

2. **Initialize()**: starts the retrieval of relevant components:

- 2.1. **GetCriticalFaultElements()** retrieves fault elements (type: *StructuralElement*) that is selected in the project visualizer. The element so retrieved will be a starting fault element and one of the critical elements.

```
List<CriticalFaultElement> startFaultElements =  
Visualizer.GetCriticalFaultElements(); //i.e. user selects one  
Or Visualizer.GetCriticalFaultElements(StructureId);
```

- 2.2. **GetCriticalEffectElements()** retrieves from project visualizer a list of critical effect elements (each of type: *StructuralElement*) that would be affected by a fault in the *CriticalFaultelement* selected under 2

```
List<StructuralElement> criticalElements =  
Visualizer.GetCriticalEffectElements();//i.e. user selects them  
or Visualizer.GetCriticalEffectElements(StructureId); or  
Visualizer.GetCriticalEffectElements(List<StructureElementIds>);
```

- 2.3. **GetEffectDefinitions()** retrieves sets of effect definitions and fault types for all the selected critical elements (*at once or one by one*). This calls the **EffectAndFaultDefinition** module which then generates effect thresholds and offers the possibility of defining and editing the related faults. This list of *EffectDefinitions* is then put into the **QFE_Interface** for reference.

```
List<EffectDefinition> effectdefinitions =  
EffectAndFaultDefinition.GetEffectDefinitions(criticalElements);  
or GetEffectDefinitions (StructureId, StructuralElementId);  
QFE_Interface.SetEffectDefinitions(effectdefinitions);
```

- 2.4. **GetEffectTypes()** retrieves the enumerations of all possible *EffectTypes* in the structure

```
List<EffectType> = Visualizer.GetEffectTypes();
```

- 2.5. **GetFaultTypes()** retrieves the enumerations of all possible *FaultTypes* in the structure.

```
List<FaultType> = Visualizer.GetFaultTypes();
```

- 2.6. **GetFaultDefinitions()** retrieves the lists of all possible initial *Faults* in the structure.

```
List<Fault> = Visualizer.GetFaultDefinitions();
```

- 2.7. For the list of critical elements or a given *StructureId*, **GetReferenceMemberReactions()** retrieves from the NumericalComputation Module the member reactions for which the element has been designed for (*at once or one by one*).

```
List<MemberReaction> referenceMemberReactions =  
NumericalComputation.GetReferenceReactions(List<CriticalElement>);
```

3. **Initiate FMEA:** to initiate the FMEA process, the user selects either a batch or an interactive mode of operation. In batch mode, the system iterates through the list of CriticalFaultElements and, for each fault element, through the list of its initial faults. In an interactive session, the user can steer the iteration by choosing the fault element to be analysed at a time, and can force to skip the cycles of iteration. In both cases, the system calls

```
AnalyzeFaultElement(StructureId, CriticalFaultElement);
```

Which in turn calls:

```
AnalyzeFault(StructureId, IterationLevel);
```

- 3.1. This first calls the method *AddFaults(StructureId, Faults)*. The module **NumericalComputation** associates the given Faults to the respective elements and returns *StructureId* of the modified structure (*or a Boolean*).

- 3.2. The module **NumericalComputation** checks whether the given Structure has become kinetically unstable and returns list of *StructureId*, i.e. the (sub) systems that became unstable, if any.

```
NumericalComputation.IsStable(StructureId);
```

- 3.3. If step 3.2 returned an unstable (sub)-structure, the TragwerkFMEA module records the result as a system level effect in Table 1 and ends the iteration. The System continues the analysis with the next Fault/FaultElement after restoring the original undisturbed status. In an interactive mode, the user can select a different CriticalFaultElement and continue the process.

Otherwise, if the structure remains stable, with the initial element fault set, *GetNumericalInfluences()* retrieves all resultant changes in the member reactions for all CriticalEffectElements and all relevant locations and relevant reaction types.

```
numericalInfluences =  
NumericalComputation.ComputeNumericalInfluences(StructureId);
```

- 3.4. ActualMemberReactions are then computed as the cumulative sum of the incremental numericalInfluences thus retrieved and the reference member reactions.

```
actualmemberreactions = updateMemberReactions(numericalInfluences);
```

- 3.5. The **QFE_Interface** is then called upon to deduce the effects that are associated with the ActualMemberReactions, which returns list of Effects with new faults, if any. This could be check each element one by one, or for all elements at once.

```
List<Effect> newEffects =
  QFE_Interface.ComputeEffectsAndFaults(actualMemberReactions);
```

It is then checked if the iteration cycle resulted in any new faults. The iteration cycle terminates if no new faults occurred.

```
Bool newFaultsOccured = QFE_Interface.isNewFaults()
```

- 3.6. The TragwerkFMEA module initializes **TableEditor** and then inserts the data retrieved from the **QFE_Interface** and the ActualMemberReactions into the relevant tables. The TragwerkFMEA module must be in a position to modify structure of the tables in **TableEditor**, and add or delete rows in them (tables 1 – 2 shown below). **TableEditor** can, from the outset, initialize these standard tables and set the proper data relations between them. In this case, the operations in the TragwerkFMEA module will only include manipulation of the data.

```
TableEditor.AddRowsToTable1(List of data objects);
TableEditor.AddRowsToTable2(List of data objects);
```

In Table 1 of the Table Editor, the following data objects that are attributes of the StructuralElement acting as the CriticalFaultElement will be inserted into the respective columns:

No.: row number, auto-incremented for each new CriticalFaultElement.

GUID: globally unique ID of the structural element

Name: name of the structural element

Function: structural function of the element

Description: characterizes the structural element

Fault Type (possible): FaultInfo describing the fault

Possible cause: possible cause of the FaultType

Local Effect: local effect associated with the FaultType

System level effects: an entry that depends on the final outcome of the FMEA run. This will be inserted at the end of the cycles of iteration.

The remaining column value should be manually completed.

A row in Table 2 of the FMECA tables consists the following data objects

Fault #: An auto-incremented row number indicating the row # in Table 1 it corresponds to.

Iteration Level: the number of iteration cycle the input belongs to. Iteration level will be counted starting with 1

Element ID: ID of the concerned StructuralElement

Element Name: Name of concerned StructuralElement

Reaction Type: Reaction type that is causing the effect

Location: point of action of the reaction

Actual value: current magnitude of the MemberReaction

% change: deviation of the current magnitude of the MemberReaction from its reference value

Effect (local): EffectInfo of the Effect caused by the Reaction

Resultant Fault: FaultInfo of the resultant fault

Comment: to be manually entered by the user.

- 3.7. The *Visualizer* should then produce a color-coded display for visual inspection of the changes in MemberReactions and their effects (using data in Table 3 below). At the same time, the user may also inspect the tabulated data and edit some entries.

```
Visualizer.DisplayEffects(newEffects);  
TableEditor.DisplayTable();
```

At the end of inspection, the user may let the iteration continue or stop it altogether. The graphical user interface must provide the necessary dialog.

- 3.8. If step 3.5 returned new faults, the TragwerkFMEA module calls the **NumericalComputation** to modify the structure with the new set of faults, and repeats the above procedures until no new faults occur, or the structure becomes unstable or the user stops the iteration. The TragwerkFMEA module must keep track of the level of iterations (tables 2).
- 3.9. At the end of the iteration, the user would be able to inspect the resulting tables, and edit the FMECA table (Table 1). *TableEditor*.DisplayTable() should display the auto-generated FMEA tables for manual edition.
- 3.10. In an interactive session, the user can, if desired, choose from the list of existing CriticalFaultElements a new starting fault element, or a completely new constellation of CriticalFaultElement and a set of CriticalEffectElements (Reinitialization), and go through the whole process.
4. In addition, the module should contain methods that enable interaction with the other modules to set or get relevant parameters.

- a. *GetFaults (StructureId, StructuralElementId)*: The module EffectAndFaultDefinition returns Faults associated to a given StructuralElement, i.e. a list of (StructuralElementId, Location, FaultType).

6 Data Structures

6.1 Conceptual Classes

The conceptualization of the task is presented in D6 Models and provides the basis for the data structures in the implementation. Figure 8 shows a conceptual class diagram of critical elements and objects related to them.

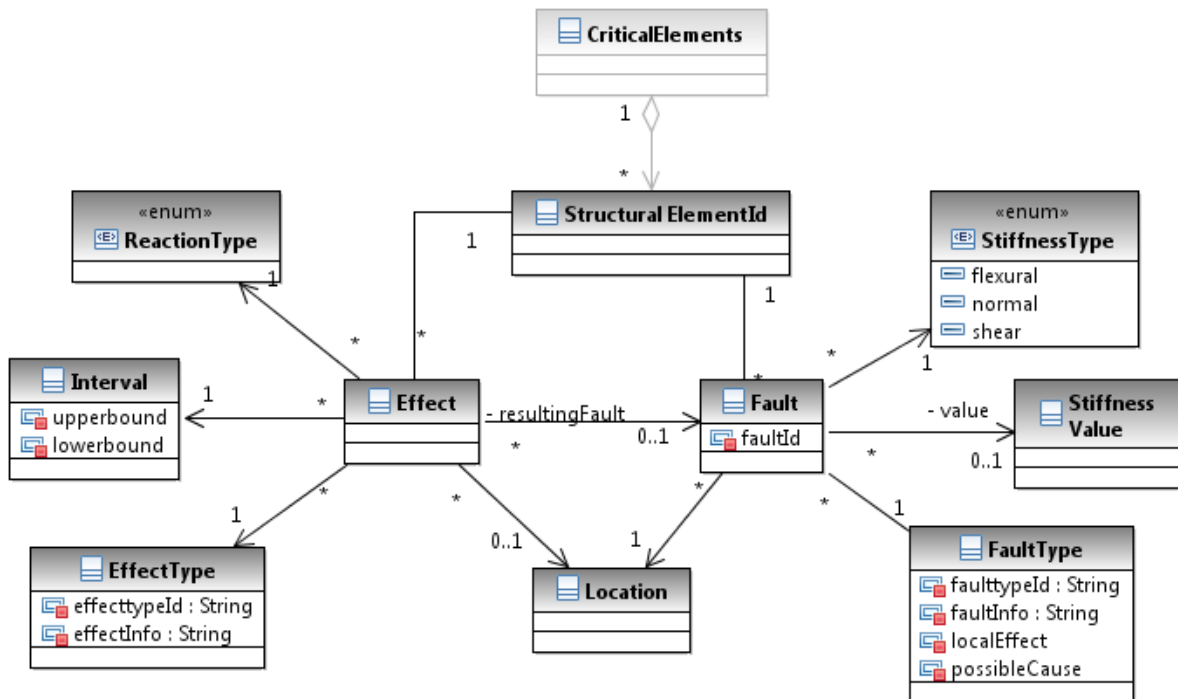


Figure 8: Conceptual class diagram

6.2 Data Exchange with Kassel Modules

1. *GetCriticalFaultElements()* returns a set of *StructuralElement* objects.

Input parameter(s)	User selects a number of <i>CriticalFaultElement</i> objects from a display in the Visualizer. Alternatively, unique IDs of the StructuralElements.
returns	A list of <i>CriticalFaultElement</i> objects. These are <i>StructuralElement</i> objects to each of which a set of initial faults are associated.

2. *GetCriticalEffectElements()* returns a list of *StructuralElement* objects.

A *StructuralElement* object has the attributes: *StructuralElementID* i.e. a globally uniqueID, *name*, *description* and *function*.

Input parameter(s)	User selects a number of <i>StructuralElement</i> objects from a display in the Visualizer. Alternatively, a list of unique IDs of the <i>StructuralElements</i> .
returns	A list of <i>StructuralElement</i> objects as <i>CriticalEffectElements</i>

3. *GetEffectDefinitions()* returns a list of *EffectDefinition* objects for each *CriticalElement* object.

Input parameter(s)	a list of <i>StructuralElement</i> objects as <i>CriticalElements</i>
returns	A list of <i>EffectDefinition</i> objects for each <i>CriticalElement</i> object

EffectDefinition has the attributes *StructuralElementID*, *ReactionType*, *Location* and List of *Effects*. *Effect* has an *Interval*, represented by lower and upper bounds of magnitude of the reaction, *EffectType*, *FaultType*, *StiffnessType* and *StiffnessValue*. These data would be collected only once and sent to QFE as references (*SetEffectDefinitions()* delivers these data to the QFE_interface).

4. *GetReferenceReactions()*

Input parameter(s)	a list of <i>StructuralElement</i> objects as <i>CriticalElements</i>
returns	A list of reference <i>MemberReaction</i> objects, each of which has the attributes: <i>StructuralElementID</i> , <i>ReactionType</i> , <i>Location</i> and <i>magnitude</i> (a real value).

5. *GetEffectTypes()*

Input parameter(s)	
returns	A list of <i>EffectType</i> objects, each of which has the attributes: <i>typeID</i> and <i>description</i> (both strings).

6. *GetFaultTypes()*

Input parameter(s)	
returns	A list of <i>FaultType</i> objects, each of which has the attributes <i>TypeId</i> , <i>FaultInfo</i> , <i>LocalEffect</i> and, <i>PossibleCause</i>

7. *GetNumericalInfluences()*

Input parameter(s)	a list of <i>CriticalFaultElement</i> objects with Faults
returns	A list of <i>NumericalInfluence</i> objects, each of which has the attributes: <i>StructuralElementID</i> , <i>ReactionType</i> , <i>Location</i> , <i>magnitude</i> (a real value) and possibly <i>iterationLevel</i> (int). <i>NumericalInfluence</i> will be computed for all <i>CriticalEffectElements</i> .

8. *AddFaults()* sends a list of faults to the module **NumericalComputation** which then associates them to the respective elements and returns *StructureId* of the modified structure.

Input parameter(s)	a list of <i>StructuralElement</i> objects with Faults, each with ID of the structural element, location, fault type, and type and value of the stiffness.
returns	<i>StructureId</i> of the modified (sub)-structure

Attributes of *Fault* are *StructuralElementID*, *Location*, *StiffnessValue*, *StiffnessType*, and *FaultType*

9. *DisplayEffects()*: For display in 3D, the same faults as under 6 will be used at each iteration level. For additional color-coded display of the relative changes in MemberReactions, the data record of current iteration level (Table 2) can be sent to the *Visualizer*.

Input parameter(s)	a list of <i>StructuralElement</i> objects with Faults; alternatively, for coded display, <i>StructuralElementID</i> , <i>ReactionType</i> , <i>Location</i> , actual <i>MemberReaction</i> , % change in Reaction, <i>FaultType</i>
returns	None, viz. a display

10. *GetInitialFaultDefinitions()*

Input parameter(s)	
returns	A set of <i>Faults</i> that would be used to associate to a structural element as initial Fault for initiation of the FMEA iteration.

11. *IsStable()*: After sending *AddFaults()* the answer to this request could be a yes/no or the ID of the structural subsystem that has become unstable (if any). Unstable (sub)-system would mean termination of the iteration cycle.

Input parameter(s)	a list of <i>StructuralElement</i> objects with Faults.
returns	<i>StructureId</i> of the modified (sub)-structure

6.3 Data Exchange with QFE:

1. *ComputeEffectsAndFaults()* Computes resultant Effects and resultant Faults of CriticalEffectElements given as argument for current values of their member reactions.

Input parameter(s)	a list of <i>CriticalEffectElement</i> objects, whose member reactions have just been updated. Sets boolean flags if new Effects and/or new Faults would result.
returns	

2. *isNewFaults()* returns true if any current MemberReaction results in a new Fault as computed under 1.
3. *isNewEffects()* returns true if any current MemberReaction results in a new Effect as computed under 1.
4. *SetEffectDefinitions()* delivers the list of *EffectDefinition* objects returned from the EffectAndFaultManager to the QFE_interface.

Input parameter(s)	a list of <i>EffectDefinition</i> objects.
--------------------	--

returns	<i>Boolean</i> (true if the objects are set successfully)
---------	---

5. *GetEffects(actual MemberReactions)* sends a list of current member reactions to QFE. Each current MemberReaction consists an ID of the StructuralElement, ReactionType, Location and magnitude of the reaction. QFE then compares the magnitudes to the reference values of the repective reactions and deduces the resulting Effect. For each *actualMemberReaction* (viz. *critical structural element*) QFE then returns a set composed of EffectType, EffectInfo, and the associated Fault to the TragwerkFmea module.

Input parameter(s)	a list of <i>actual MemberReaction</i> objects.
returns	A list of <i>Effect</i> objects, one for each <i>actual MemberReaction</i> object sent

For details see the document: “QFE: Specification Document”.

6.4 Data Exchange with TableEditor

As the structure of the tables above shows, the data sets to be sent to the TableEditor are complex. Each table is a combination of strings and real numbers, which are attributes of one or more of the classes mentioned above, with relatively complex handling procedures.

Data sent to Table 1 (see above) are initial user inputs excepting “system level effects”, which is going to be the final result of the iteration with the initial fault. In this table the 1st column (No.) is a serial number auto-incremented for each initial fault.

The table for recording changes in MemberReactions (Table 2 above) will contain results of the computations of actual member reactions and the associated local effects for all the critical elements and relevant locations in each of them. In this table Fault# refers to the serial no. in Table 1, and Iteration level will be auto-incremented if the computaion is going to be repeated with new faults resulting from the preceding one(s). The user may be in a position to add comments, but the advantage of doing so is not visible now. This table is a basis for displaying the effects following each iteration in the Visualizer.

In Table 1 of the Table Editor, the following attributes of the StructuralElement acting as the CriticalFaultElement will be inserted into the respective columns:

No.: row number, auto-incremented for each new CriticalFaultElement.

GUID: globally unique ID of the structural element

Name: name of the structural element

Function: structural function of the element

Description: characterizes the structural element

Fault Type (possible): FaultInfo describing the fault

Possible cause: possible cause of the FaultType

Local Effect: local effect associated with the FaultType

System level effects: an entry that depends on the final outcome of the FMEA run. This will be inserted at the end of the cycles of iteration.

The remaining column value should be manually completed.

A row in Table 2 of the FMECA tables consists the following:

Fault #: An auto-incremented row number indicating the row # in Table 1 it corresponds to.

Iteration Level: the number of iteration cycle the input belongs to. Iteration level will be counted starting with 1

Element ID: ID of the concerned StructuralElement

Element Name: Name of concerned StructuralElement

Reaction Type: Reaction type that is causing the effect

Location: point of action of the reaction

Actual value: current magnitude of the MemberReaction

% change: deviation of the current magnitude of the MemberReaction from its reference value

Effect (local): EffectInfo of the Effect caused by the Reaction

Resultant Fault: FaultInfo of the resultant fault

Comment: to be manually entered by the user.

AddTowsToTable1(row); AddTowsToTable2(row);

Input parameter(s)	<i>row:</i> with entries for each of the columns in table 1, resp. table 2
returns	

Finally, the rest of the columns are then edited manually. At the end of the computational session, the user must be in a position to add other row entries for which values come from other sources (see Deliverable D3 “Richtlinien”)

7 A Session with the Structure-FMEA Tool

In the following, we illustrate a session using an artificial example.

7.1 Starting StructureFMEA

Starting “StructureFMEA.exe” (double-clicking) will open the main window as shown in Figure 9. Currently the menu point “File” holds the menu items “New” and “Close” only.

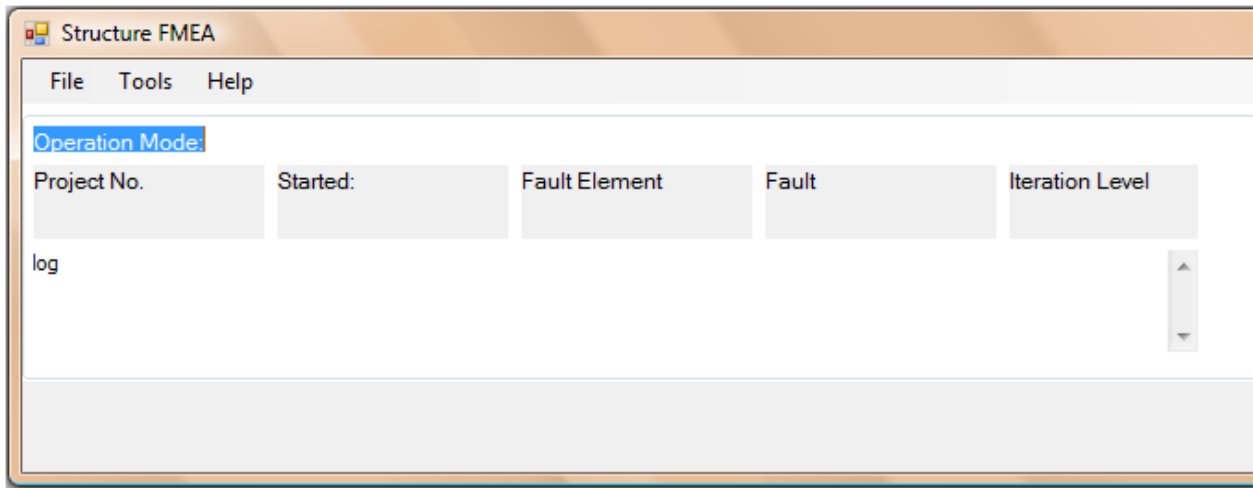


Figure 9 Structure FMEA main Window

Choose File/New will open a dialog shown in Figure 10 to enter a Project number (entry is required, otherwise the button “Start” remains inactive). Optionally one can enter a description of the project.

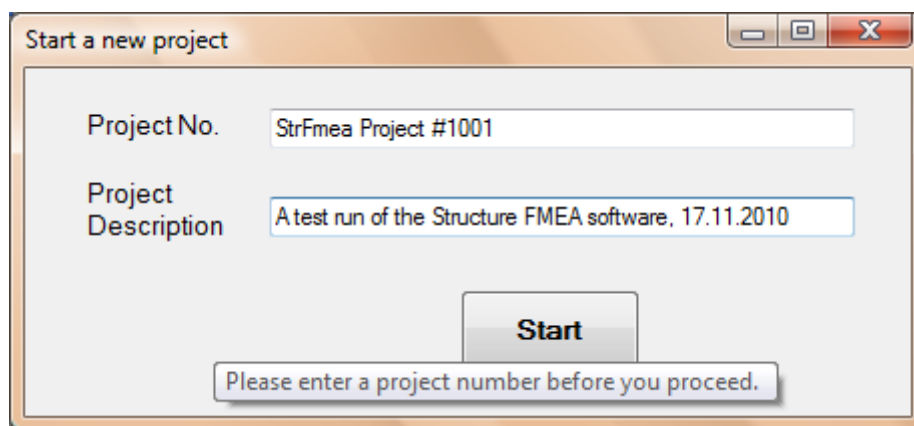


Figure 10 Clicking File/New opens this new project dialog.

7.2 Opening the Structural Concept

After clicking the button “Start”, the user will be prompted to open a file containing the Structural Concept. This is presumed to have the extension “.cet”. Otherwise, an error will be displayed that the file does not contain a proper Structural Concept.

7.3 Selecting Critical Elements

After the Structural concept has been opened, the user is prompted 1st to choose critical fault elements, and in a second round, to choose the critical effect elements from within the Visualizer. Here it is only a mock up selection. Any way, a prepared set of fault, resp. effect elements will be returned with all fault and effect definitions necessary for the demo.

7.4 Modes of Operation

After the critical elements have been retrieved successfully in background, the user is prompted to choose between batch mode and interactive mode of operation (Figure 11). In

batch mode, the FMEA analysis runs with the selected sets of Faults/FaultElements and Critical Effect Elements without user intervention.

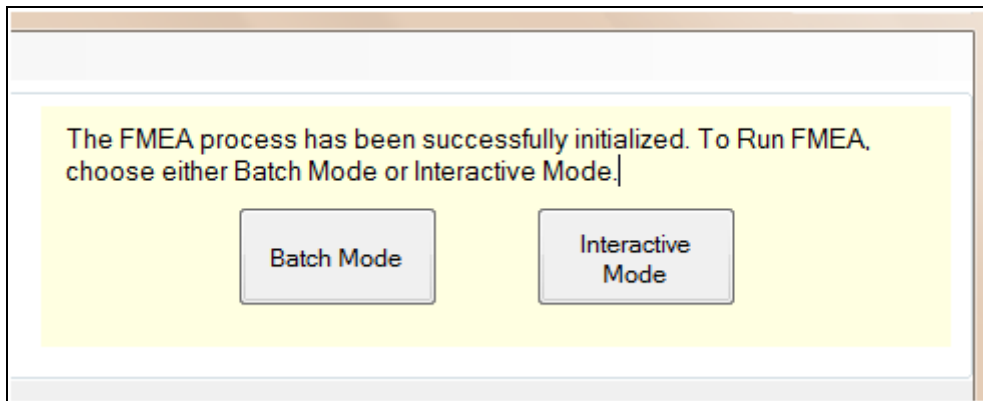


Figure 11: The user is prompted to choose a batch or an interactive mode of operation

Interactive Mode of Operation

a. Selecting a Fault Element for Analysis

In an interactive mode, the user can select one FaultElement at a time from the set collected during the initialization phase (Figure 12) and run the analysis interactively. At the end of both modes of operation, the analysis results are saved automatically.

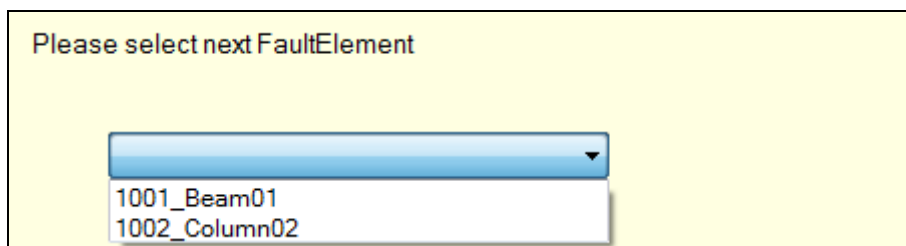


Figure 12: Selection of a fault element for analysis

b. Selecting form of display

In an interactive session, the user is prompted if he/she wants to view the intermediate results at the end of every iteration cycle. For that, the user can choose between the visualizer and the table editor. The selection remains displayed until he/she presses the OK button to close the dialog. Then the dialog in Figure 13 reappears so that the user can choose another form of display. After enough inspection, the user can press "No Display" to continue with the analysis.

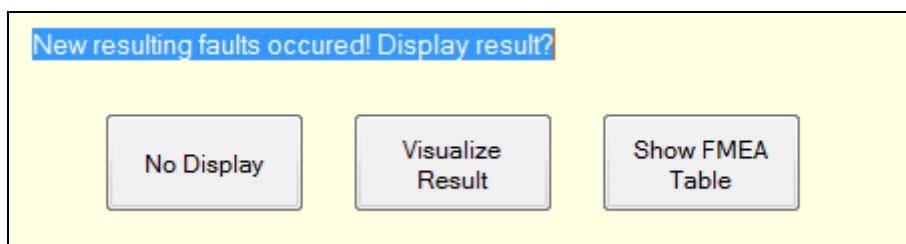


Figure 13: Selecting how to display result of an analysis

For this demo, there is no proper display of effects in the visualizer, except entries of resultant effects in a text box. A data structure exchanged with the Visualizer contains (element id, reaction type, location, current reaction value, change in %, resulting fault).

In the table editor, attributes of the structural element acting as the CriticalFaultElement will be inserted into the respective columns in Table 1 of the table editor (Figure 14). Table 2 is devised as a pivot table upon rows in Table1, and contains results of the computations of actual member reactions and the associated local effects for all the critical effect elements and relevant locations in each of them.

GUID	Name	Function	Fault Type	Possible Cause	A	Remedial Action	Local Effect	System Level Effects	B	Detection Means	E	RPN	C
1001	Beam	support	Reduction in stiffness of	Reduced section, mechanical damage or material deterioration	0		Reduced ca	Resulted in unstable	0		0	0	
1001	Beam	support	Overstressing	Loads are more that what the section is designed for	0		Fails due to	Resulted in unstable	0		0	0	

Figure 14: View in Table Editor. Each fault of a Structural element produces one row in table1.

c. Ending analysis of a fault/ of a fault element

During an interactive session, the user has the option to prematurely end the iteration cycles of a given fault, for which he/she is prompted as in Figure 15. Otherwise, the iteration cycle ends if no new faults occur, or the fault results in some system level instability.

In case the selected fault element contains a number of initial faults, the user can opt to skip analysing the remaining faults in a dialog shown in Figure 16.

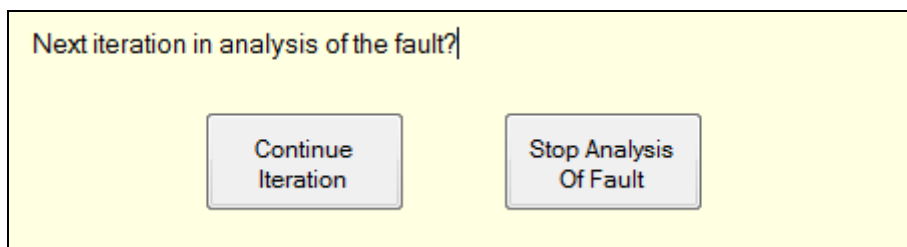


Figure 15: The user can opt to end cycles of iteration with one fault

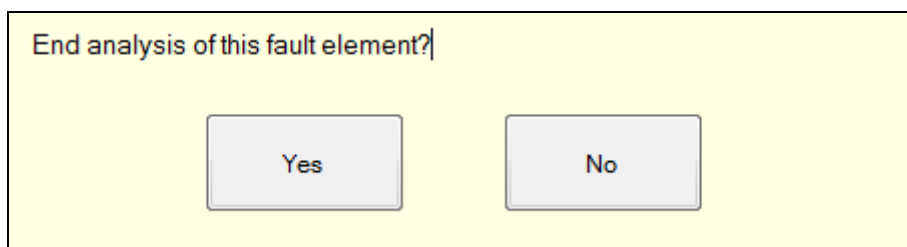


Figure 16: The user can skip analysis of remaining faults of a critical fault element

After finishing analysis of the selected FaultElement, the user is prompted if he/she wants to analyze another fault element in the set retrieved during the initialization phase (Figure 17).

If the user chooses “Yes”, he/she will be presented the dialog shown in Figure 12 again. Otherwise, he/she is prompted with a reinitialization dialog shown in Figure 18.

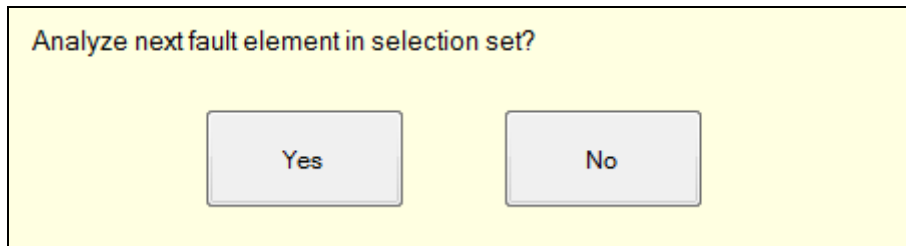


Figure 17: A user prompt (in interactive mode) for selection of next fault element

e. Reinitialization

The dialog in Figure 18, then, prompts the user for reinitialization if desired. The user can select to reinitialize (retrieve anew) either fault elements only (the effect elements will be reused), or effect elements only (the fault elements will be reused), or retrieve both anew. The procedure can then be repeated with the new sets. If the user opts for “No Initialization”, the FMEA process terminates, the results in the table editor are saved automatically.

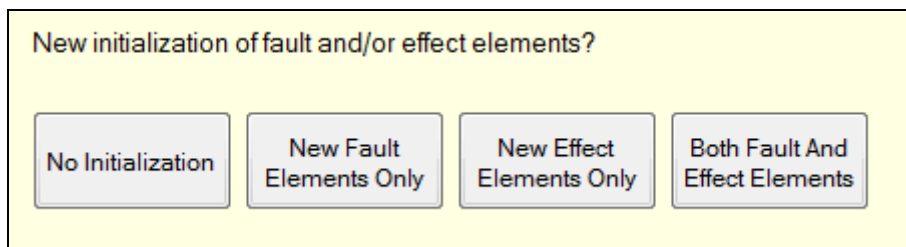


Figure 18: A dialog for reinitialization of the FMEA process with new Fault and/or Effect elements

In interactive mode, the user is also prompted if he/she wants to save the logs to a file.

7.5 Editing the Results Table

At the end, the Visualizer and the Table Editor are displayed for inspection. Figure 19 shows the details of analysis results in a table editor.

GUID	Name	Function	Fault Type	Possible Cause	A Remedial Action	Local Effect	System Level Effects	B Detection Means	E RPN	Control
1001	Beam01	support	Reduction in stiffness of structural element	Reduced section, mechanical damage c	0	Reduced c	Resulted in unstable	0	0	0
1001	Beam01	support	Overstressing	Loads are more that what the section is	0	Fails due to	Resulted in unstable	0	0	0
1002	Column02	support	Reduction in stiffness of structural element	Reduced section, mechanical damage c	0	Reduced c	Resulted in unstable	0	0	0

Iteration Level	Element ID	Element Name	Reaction Type	Locatio	Actual Value	Percentage Change	Effect	Resultant Fault	Comment
1	1003	Beam03	SHEAR_Z		102.98	21.1	Carrying capacity of the element is exhausted	Yielding	
1	1004	Column04	SHEAR_Z		106.47	25.3	Carrying capacity of the element is exhausted	Yielding	
1	1006	Column06	SHEAR_Z		114.06	34.2	Structural element collapses	Overstressing	
2	1003	Beam03	SHEAR_Z		130.1	53.1	Structural element collapses	Overstressing	
2	1004	Column04	AXIAL		-358.93	19.6	Structural element collapses	Overstressing	
2	1004	Column04	SHEAR_Z		119.81	41	Structural element collapses	Overstressing	
2	1004	Column04	TORSIONAL		92.19	84.4	Carrying capacity of the element is exhausted	Yielding	
2	1005	Beam05	SHEAR_Z		104.28	22.7	Carrying capacity of the element is exhausted	Yielding	
2	1006	Column06	SHEAR_Z		144.17	69.6	Structural element collapses	Overstressing	
2	1006	Column06	TORSIONAL		105.77	111.5	Structural element collapses	Overstressing	

Rows: 3

Figure 19 Clicking in one of the rows in Table 1 shows the details of the analysis in lower table

The Table Editor can also be used as a stand alone application, that enables the user to open existing FMEA data, editing and saving under a different file, and printing.

8 Benefits of Using the Tool

< To be done by Kassel >