Requirement Management Tool rmtoo

Introduction

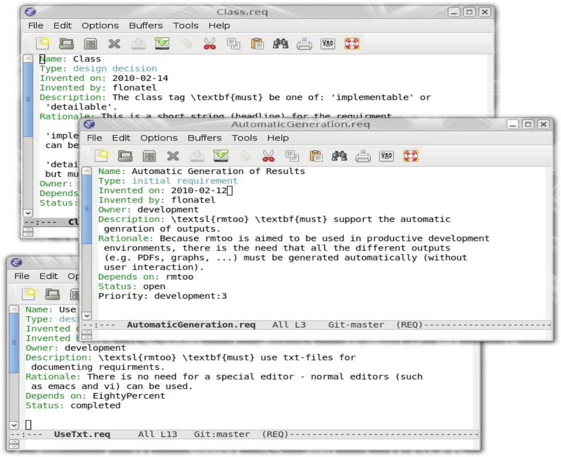
by flonatel GmbH & Co. KG

Version 3

Content

1.Introduction

Overview



Class.req

```
Name: Class
Type: design decision
Invented on: 2010-02-14
Invented by: flonatel
Description: The class tag \textbf{must} be one of: 'implementable' or
 'detailable'.
Rationale: This is a short string (headline) for the requirment
 'imple
 can be
 'detai
 but mu
Owner:
Depends
Status:
--:---  Cl
```

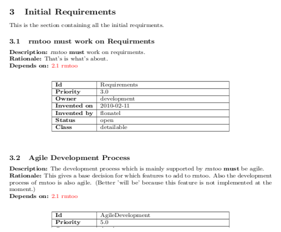
AutomaticGeneration.req

```
Name: Automatic Generation of Results
Type: initial requirement
Invented on: 2010-02-12
Invented by: flonatel
Owner: development
Description: \textsl{rmtoo} \textbf{must} support the automatic
 genration of outputs.
Rationale: Because rmtoo is aimed to be used in productive development
 environments, there is the need that all the different outputs
 (e.g. PDFs, graphs, ...) must be generated automatically (without
 user interaction).
Depends on: rmtoo
Status: open
Priority: development:3
--:---  AutomaticGeneration.req    All L3    Git-master  (REQ)----------------------
```

```
Name: Use
Type: desi
Invented 
Invented 
Owner: development
Description: \textsl{rmtoo} \textbf{must} use txt-files for
 documenting requirments.
Rationale: There is no need for a special editor - normal editors (such
 as emacs and vi) can be used.
Depends on: EightyPercent
Status: completed

--:---  UseTxt.req    All L13   Git:master  (REQ)----------------------------------
```

rmtoo



3 Initial Requirements

This is the section containing all the initial requirments.

3.1 rmtoo must work on Requirments

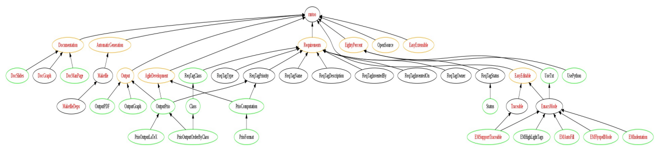
Description: *rmtoo* **must** work on requirments.
Rationale: That's is what's about.
Depends on: 2.1 rmtoo

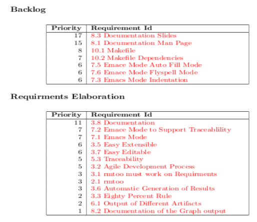
Id	Requirements
Priority	3.0
Owner	development
Invented on	2010-02-11
Invented by	flonatel
Status	open
Class	detailable

3.2 Agile Development Process

Description: The development process which is mainly supported by *rmtoo* **must** be agile.
Rationale: This gives a base decision for which features to add to rmtoo. Also the development process of rmtoo is also agile. (Better 'will be' because this feature is not implemented at the moment.)
Depends on: 2.1 rmtoo

Id	AgileDevelopment
Priority	5.0





Backlog

Priority	Requirement Id
17	8.3 Documentation Slides
15	8.1 Documentation Man Page
8	10.1 Makefile
7	10.2 Makefile Dependencies
6	7.5 Emace Mode Auto Fill Mode
6	7.6 Emace Mode Flyspell Mode
6	7.3 Emacs Mode Indentation

Requirments Elaboration

Priority	Requirement Id
11	3.8 Documentation
7	7.2 Emace Mode to Support Traceablility
7	7.1 Emacs Mode
6	3.5 Easy Extensible
6	3.7 Easy Editable
5	5.3 Traceability
5	3.2 Agile Development Process
3	3.1 rmtoo must work on Requirments
3	2.1 rmtoo
3	3.6 Automatic Generation of Results
2	3.3 Eighty Percent Rule
2	6.1 Output of Different Artifacts
1	8.2 Documentation of the Graph output

Input Data   Command line tool   Output Artifacts

Basic Facts (1/2)

- *rmtoo* is a minimalistic non-interactive requirements management tool
- *rmtoo* works on data stored in the file system (plain text files)
- *rmtoo* is a command line tool which reads in files and creates output
- *rmtoo* supports different output formats and artifacs

Basic Facts (2/2)

- *rmtoo* data files can be handled by standard *nix commands (emacs, vi, grep, awk, streplace, sed, ...)
- *rmtoo* runs (mostly) on the same hardware and operating system where the development takes place – no need for a dedicated machine
- *rmtoo* baselineing, backup and restore can be done by a revision control system

rmtoo is not

- *rmtoo* has no GUI
- *rmtoo* comes with no database
- *rmtoo* has no integrated editor
- *rmtoo* does not provide an UML editor
- *rmtoo* does not provide any import possibility (e. g. from a spreadsheet or a word processing document)

License

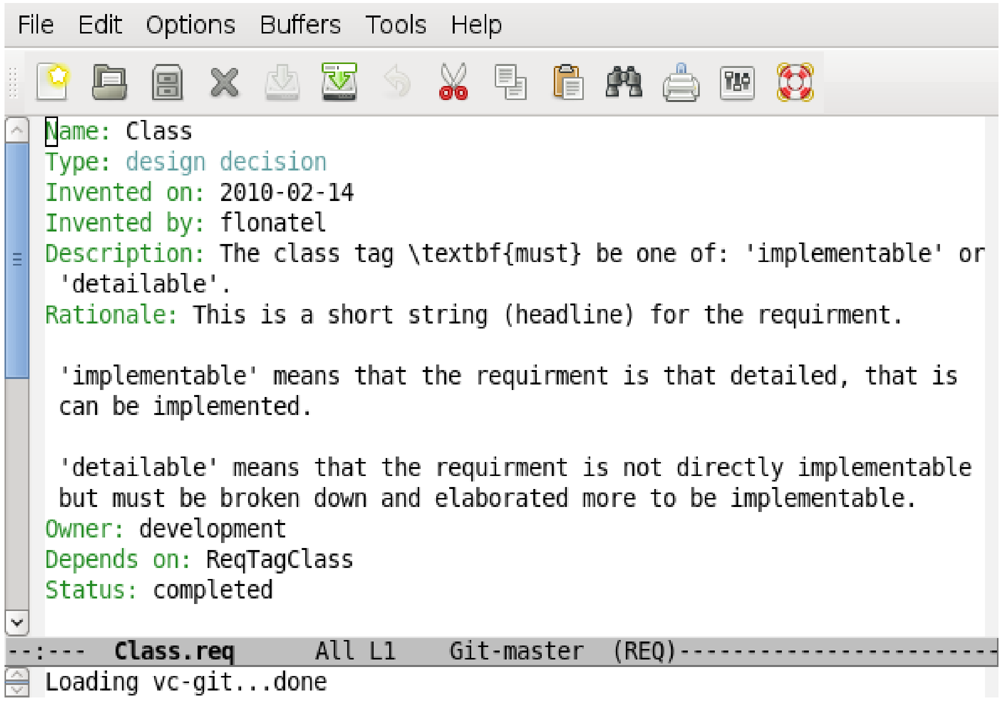
- *rmtoo* is Open Source
- *rmtoo* is free
- *rmtoo* is licensed under GPLv3
- Commercial support is available

2.Input Data Files

Data Files

- Input files are standard plain text files
- Each requirement is basically a list of key-value pairs
- Most used keys for requirement management are supported
- Files can be handled by most *nix commands (sed, streplace, awk, grep, ...)
- Revision control can be done by revision system (for some features git is needed)

Data File Example



```
Name: Class
Type: design decision
Invented on: 2010-02-14
Invented by: flonatel
Description: The class tag \textbf{must} be one of: 'implementable' or
 'detailable'.
Rationale: This is a short string (headline) for the requirment.

 'implementable' means that the requirment is that detailed, that is
 can be implemented.

 'detailable' means that the requirment is not directly implementable
 but must be broken down and elaborated more to be implementable.
Owner: development
Depends on: ReqTagClass
Status: completed
--:---  Class.req      All L1     Git-master  (REQ)-----------------------
Loading vc-git...done
```

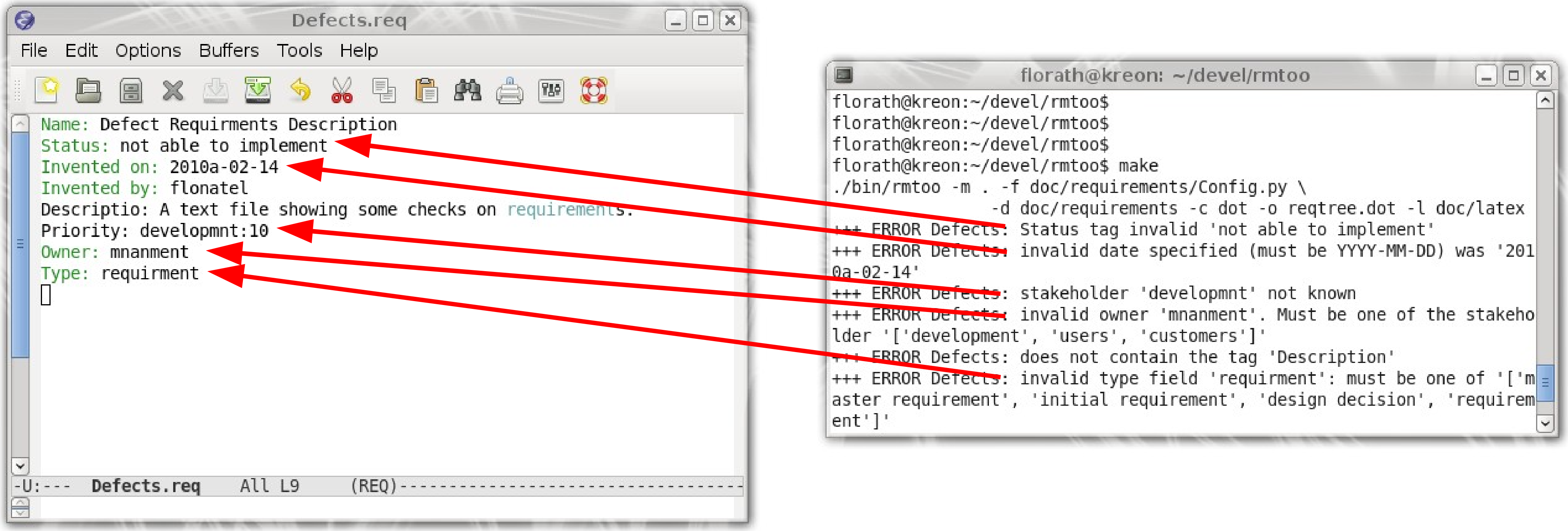
- Simple *key: value* notation
- Space in first column: extend value
- Keys are fixed (predefined)
- Values are checked when possible
- Editable with standard text editor

3. Checks on Requirements

Checks on Requirements

- Because there is no built-in editor, consistency checks must be done
- Checks include:
  - Syntax / Format checks e.g. for date fields
  - Type checks: some fields are allowed to contain only a limited set of (key-)words
  - Typo checks for e.g. stakeholders
  - Dependency checks

Example: Checks


Defects.req
File Edit Options Buffers Tools Help
-U:--- Defects.req All L9 (REQ)
florath@kreon: ~/devel/rmtoo

```
Name: Defect Requirments Description
Status: not able to implement
Invented on: 2010a-02-14
Invented by: flonatel
Descriptio: A text file showing some checks on requirements.
Priority: developmnt:10
Owner: mnanment
Type: requirment
```

```
florath@kreon:~/devel/rmtoo$
florath@kreon:~/devel/rmtoo$
florath@kreon:~/devel/rmtoo$
florath@kreon:~/devel/rmtoo$ make
./bin/rmtoo -m . -f doc/requirements/Config.py \
                -d doc/requirements -c dot -o reqtree.dot -l doc/latex
+++ ERROR Defects: Status tag invalid 'not able to implement'
+++ ERROR Defects: invalid date specified (must be YYYY-MM-DD) was '201
0a-02-14'
+++ ERROR Defects: stakeholder 'developmnt' not known
+++ ERROR Defects: invalid owner 'mnanment'. Must be one of the stakeho
lder '['development', 'users', 'customers']'
+++ ERROR Defects: does not contain the tag 'Description'
+++ ERROR Defects: invalid type field 'requirment': must be one of '['m
aster requirement', 'initial requirement', 'design decision', 'requirem
ent']'
```

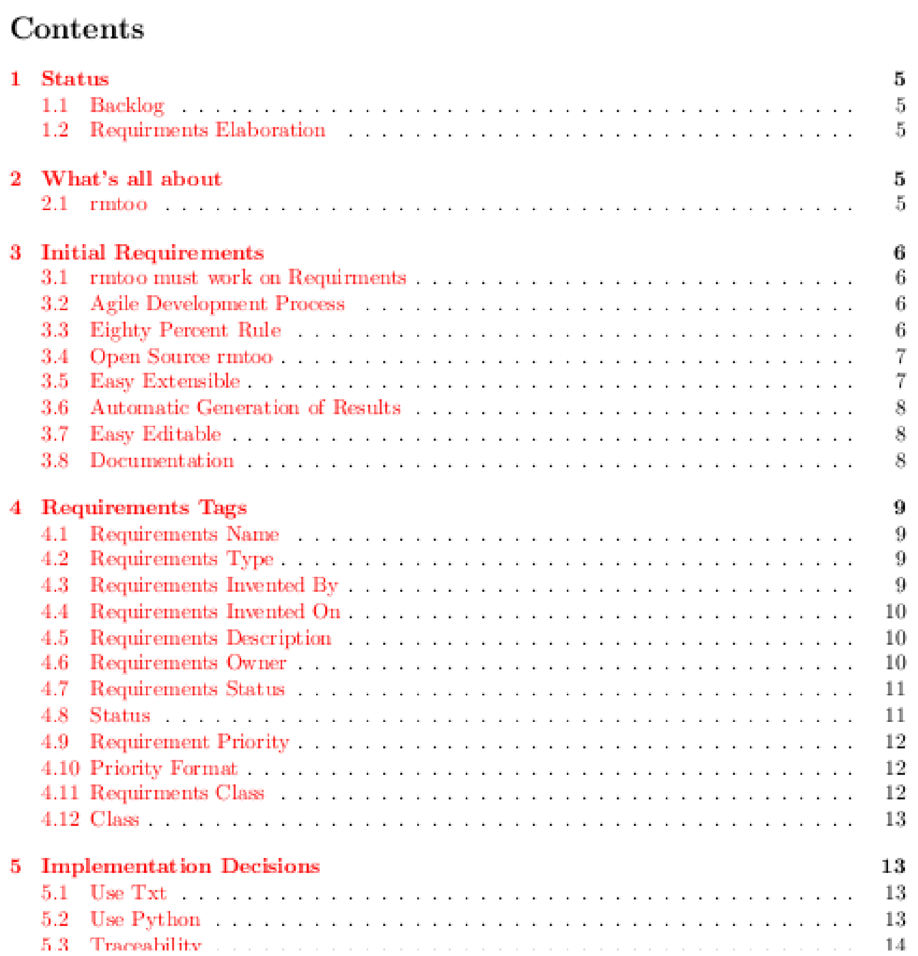
4.Output Artifacts

Requirements Document – Requirements Dependency Graph – Project Backlog – Project Elaboration List – Requirements Count Stats

Output: Requirements Document

- *rmtoo* can create a requirements document containing all requirements
- Output intermediate format of requirements is LaTeX using hyperref
- Resulting documents can be e.g. PDF and HTML
- Links in table of contents and dependencies available in PDF and HTML
- Arbitrary text can be added

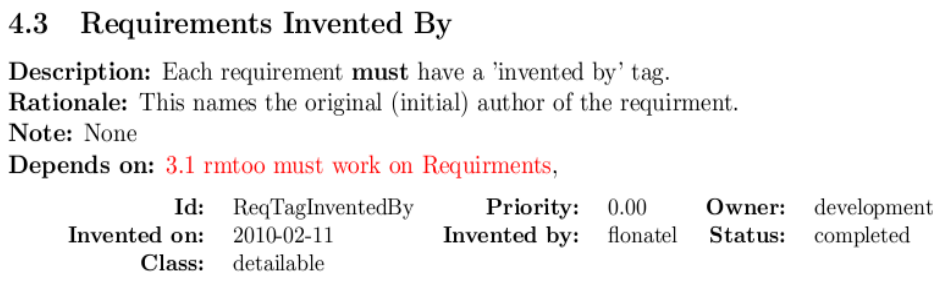
Output: Table of Contents



Contents

- Each requirement fits in it's own subsection
- Hyperlinks for fast navigation

Output: Requirement



4.3 **Requirements Invented By**

Description: Each requirement **must** have a 'invented by' tag.
Rationale: This names the original (initial) author of the requirment.
Note: None
Depends on: 3.1 rmtoo must work on Requirments,

Id:	ReqTagInventedBy	**Priority:**	0.00	**Owner:**	development
Invented on:	2010-02-11	**Invented by:**	flonatel	**Status:**	completed
Class:	detailable				

- Each requirement fits in it's own subsection

  All key-values are available

- Hyperlinks to dependencies for fast navigation

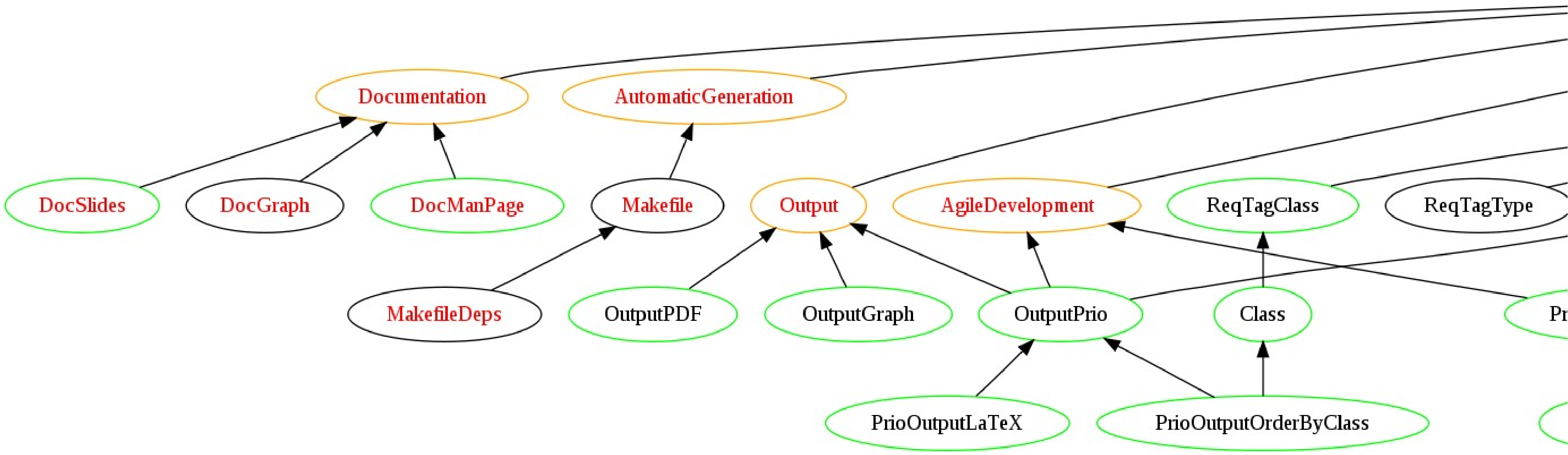
4.Output Artifacts

Requirements Document – Requirements Dependency Graph – Project Backlog – Project Elaboration List – Requirements Count Stats

Output: Dependency Graph

- *rmtoo* can create a requirement dependency graph
- Simple to visualize dependencies of requirements
- Colorized status information. Example: red font means open, black font completed

Output: Dependency Graph Example (Part)



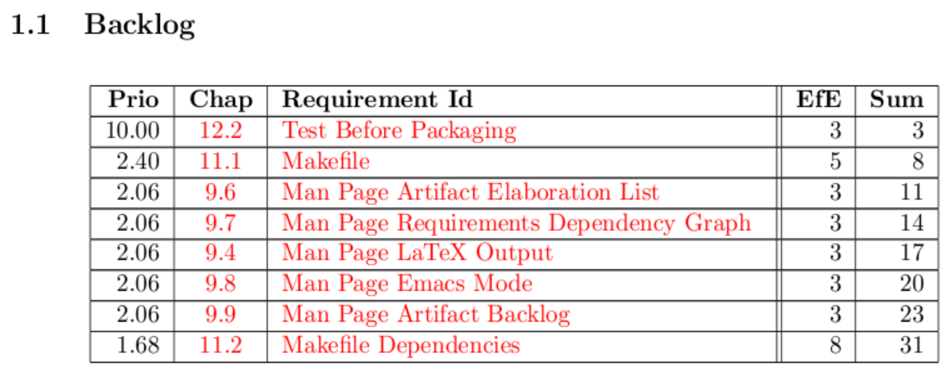
4. Output Artifacts

Requirements Document – Requirements Dependency Graph – **Project Backlog** – Project Elaboration List – Requirements Count Stats

Output: Project Backlog

- *rmtoo* can create the project backlog as used in SCRUM
- Project backlog contains all elaborated requirements (which means they can be implemented)
- Project backlog is the ToDo list for the developers
- Requirements are sorted by priority

Output: Project Backlog Example



1.1 Backlog

Prio	Chap	Requirement Id	EfE	Sum
10.00	12.2	Test Before Packaging	3	3
2.40	11.1	Makefile	5	8
2.06	9.6	Man Page Artifact Elaboration List	3	11
2.06	9.7	Man Page Requirements Dependency Graph	3	14
2.06	9.4	Man Page LaTeX Output	3	17
2.06	9.8	Man Page Emacs Mode	3	20
2.06	9.9	Man Page Artifact Backlog	3	23
1.68	11.2	Makefile Dependencies	8	31

- Prioritized list of requirements
- Hyperlinks for fast navigation
- Included Effort estimation
- Embedded in the PDF / HTML document

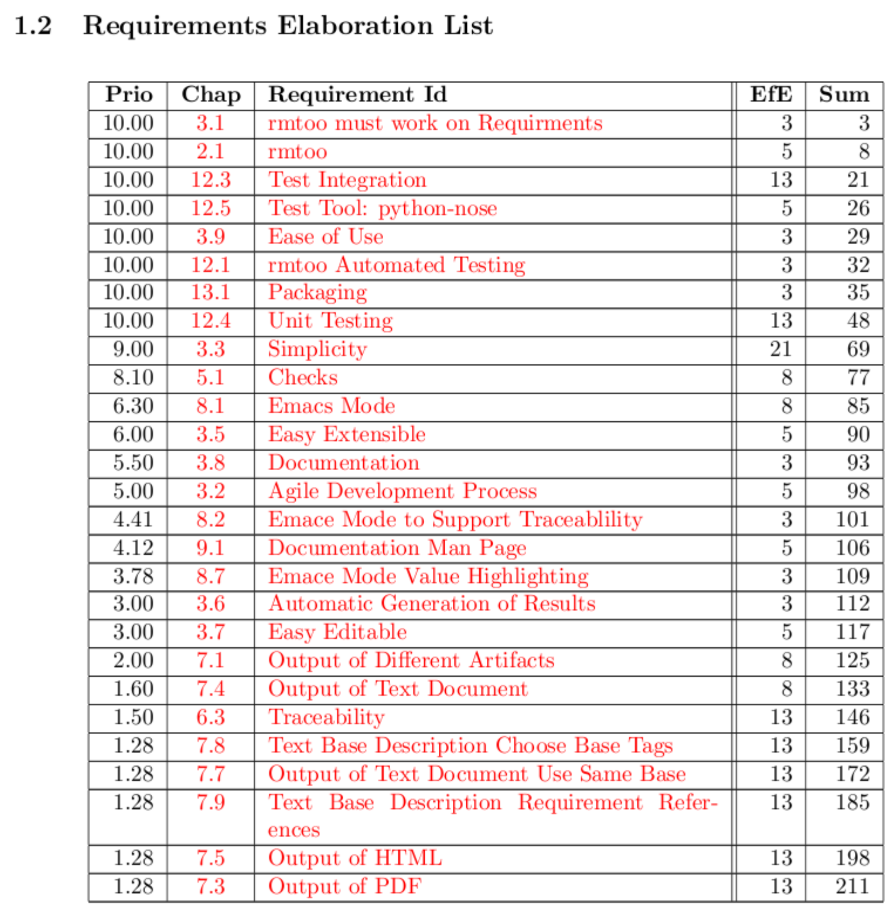
4. Output Artifacts

Requirements Document – Requirements Dependency Graph – Project Backlog – **Project Elaboration List** – Requirements Count Stats

Output: Project Elaboration List

- *rmtoo* can create a list of all requirements that must be further elaborated
- Elaboration List is the ToDo list for the SCRUM master
- Requirements are sorted by priority

Output: Project Elaboration List Example



1.2 Requirements Elaboration List

Prio	Chap	Requirement Id	EfE	Sum
10.00	3.1	rmtoo must work on Requirments	3	3
10.00	2.1	rmtoo	5	8
10.00	12.3	Test Integration	13	21
10.00	12.5	Test Tool: python-nose	5	26
10.00	3.9	Ease of Use	3	29
10.00	12.1	rmtoo Automated Testing	3	32
10.00	13.1	Packaging	3	35
10.00	12.4	Unit Testing	13	48
9.00	3.3	Simplicity	21	69
8.10	5.1	Checks	8	77
6.30	8.1	Emacs Mode	8	85
6.00	3.5	Easy Extensible	5	90
5.50	3.8	Documentation	3	93
5.00	3.2	Agile Development Process	5	98
4.41	8.2	Emace Mode to Support Traceablility	3	101
4.12	9.1	Documentation Man Page	5	106
3.78	8.7	Emace Mode Value Highlighting	3	109
3.00	3.6	Automatic Generation of Results	3	112
3.00	3.7	Easy Editable	5	117
2.00	7.1	Output of Different Artifacts	8	125
1.60	7.4	Output of Text Document	8	133
1.50	6.3	Traceability	13	146
1.28	7.8	Text Base Description Choose Base Tags	13	159
1.28	7.7	Output of Text Document Use Same Base	13	172
1.28	7.9	Text Base Description Requirement References	13	185
1.28	7.5	Output of HTML	13	198
1.28	7.3	Output of PDF	13	211

- Prioritized list of requirements
- Hyperlinks for fast navigation
- Included Effort estimation
- Embedded in the PDF / HTML document

4.Output Artifacts

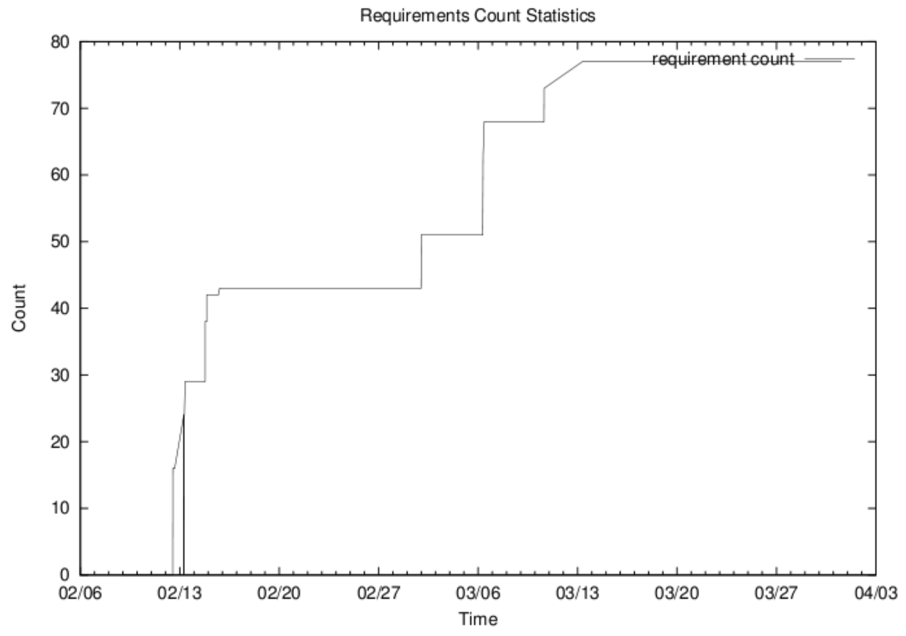
Requirements Document – Requirements Dependency Graph – Project Backlog – Project Elaboration List – Requirements Count Stats

Output: Requirements Count Stats

FLONATEL

- *rmtoo* can create a cvs file of the whole history for the number of requirements
- Precondition: history must available in git repository
- Can be preprocessed by spreadsheet or gnuplot

Output: Requirements Count Stats Example



- Number of requirements at a given point of time
- Embedded in the PDF / HTML document

5. Future

Future / Plans

- Some features are missing but planned for the next releases
  - Traceability
  - Better support writing requirements in Emacs mode
- Community, User and Customer driven

Thank you!

Copyright

This document is distributed under the creative commons license 'Attribution-Noncommercial-No Derivative Works 3.0 Germany'

© 2010 flonatel GmbH & Co. KG