Got Zest?
eXtreme Management of projects
in Plone

Maurits van Rees
Student number 0545701
Class inf5a/dua6a
Hogeschool Rotterdam

May 12, 2006
Preface

This is the report for my practice (or internship) at Zest Software in Hoogvliet, the Netherlands. My main task there was to make the eXtremeManagement tool ready for use and release it to the general public. Some extensions were welcome as well of course. The plan succeeded.

eXtremeManagement is a Plone product that helps in developing using the Extreme Programming \(^1\) methodology. It helps steer a project to success by a focus on planning, on estimating and on keeping a log of your hours worked. The colleagues and clients at Zest like it.

eXtremeManagement is now being used by Zest and several other companies. There are lots of ideas for improvement so this story is far from finished.

I like to thank a few people:

- my parents for their support;
- my brother Reinout for introducing me at Zest Software;
- Hans Manni for getting my rear in gear;
- Jean-Paul and Esther Ladage for giving me a chance to prove myself;
- Ahmad for starting the eXtremeManagement project with Jean-Paul;
- Mirella, Joris, Daniel and aforementioned colleagues for making Zest Software a very nice place to work at;
- Achtung for enthusiastically wagging his tail and not biting me (much).

---

# Contents

1 Introduction 1
   1.1 Zest Software ............................................. 1
   1.2 Python, Zope and Plone ................................. 1
   1.3 eXtreme Programming ................................. 2
   1.4 eXtremeManagement ...................................... 3
   1.5 Improvement possibilities ............................. 4

2 Release 1.0: getting it ready 6
   2.1 Workflows ................................................ 6
   2.2 Task estimates .......................................... 12
   2.3 Overview of bookings ................................... 15

3 Releasing a Plone product 21
   3.1 Introduction ............................................. 21
   3.2 Product versions .......................................... 21
   3.3 Steps on plone.org ....................................... 23
   3.4 Mailing lists ............................................ 24
   3.5 Announcement ........................................... 25

4 Release 1.1: new features proposed 27
   4.1 Auto-assign new tasks .................................. 27
   4.2 Better progress indicators ............................ 31
   4.3 Integration with the Poi issue tracker ............... 34
   4.4 Moving to the collective .............................. 35

5 Conclusion 37
   5.1 What is the status of the tool? ....................... 37
   5.2 Which improvements can be made to the tool? .... 37
   5.3 What happens after the practice? ................... 37

Bibliography 38

A Makefile 39

B Announcement 42
CONTENTS

C  Client work 44
   C.1  Marco Polo platform 44
   C.2  MinaRaad 44
   C.3  Philips Research ICT 45

D  Project: practice report 46

E  Colophon 50
# List of Figures

1.1 Stories at the end of iteration 1.0 ........................................ 5  
2.1 Project workflow .......................................................... 8  
2.2 Iteration workflow ....................................................... 9  
2.3 Story workflow ............................................................ 10  
2.4 Task workflow ............................................................. 11  
2.5 Task estimate ............................................................... 13  
2.6 Task class ................................................................. 13  
2.7 Tagged values of Task .................................................... 14  
2.8 The portlet ‘Project Administration’ .................................. 16  
2.9 My bookings for the month April ...................................... 17  
2.10 My booking details for the month May ............................... 18  
2.11 Bookings for the task of writing this section ..................... 19  
2.12 Python script getDailyBookings.py ................................. 20  
4.1 Stories for iteration 1.1 .................................................... 28  
4.2 Progress indicators ....................................................... 32  
D.1 eXtremeManagement report: project view ....................... 47  
D.2 eXtremeManagement report: structure iteration ............... 48  
D.3 eXtremeManagement report: content iteration ............... 49
Abstract

This report is an overview of my practice at Zest Software. It was a fun and educating time, which is hopefully noticeable when reading this report.

The practice started on November 29th 2005 and ended at April 30th 2006.

Chapter 1 on page 1 lays the foundation for this report. It introduces Zest Software and the main subject of the practice: the not yet finished eXtreme-Management tool. That is a tool for project management based on the Extreme Programming methodology. It is based on Python, Zope and Plone. A brief overview is given of these technologies.

Chapter 2 on page 6 discusses several improvements that have been made to the eXtremeManagement tool. Three subjects are highlighted: workflows, task estimation and booking overviews.

eXtremeManagement was released on plone.org. The steps taken to do that can be interesting for others as well. That is why I generalised that process and put it in a tutorial on the documentation section of that website. Chapter 3 on page 21 contains that tutorial.

Chapter 4 on page 27 proposes some new features. Automatic assigning of new tasks has already been implemented. Better progress indicators and integration with a bug tracker are still on the wish list. eXtremeManagement should also be moved from the Zest code repository to the Plone collective.

The conclusion in chapter 5 on page 37 is that the practice was successful. Moreover: I keep working at Zest Software for the rest of my study.
Chapter 1

Introduction

This chapter is meant to get the reader up to speed on several issues that are central to the practice. What kind of company is Zest Software? What technologies are they using? Why did they choose eXtreme Programming as their programming methodology? And where does the eXtremeManagement tool fit in all this? Lastly, what needs to be done in the practice to improve the eXtremeManagement tool?

1.1 Zest Software

Zest Software is a software development company based in Hoogvliet, the Netherlands, which is focused at delivering high quality web application software based on Zope and Plone, technologies that are explained below. The company was started around the year 2002 by Jean-Paul and Esther Ladage. Soon Ahmad Hadi joined the team and began working on the eXtremeManagement tool. Business has grown steadily and new coworkers joined. As of April 2006 Zest has grown to seven people and a fierce guard dog called Achtung who will wag his tail when any intruder comes near.

Zest basically makes websites using the tools mentioned below. Customers include Philips Research, Triple-P and MilieuDefensie (Friends of the Earth Netherlands).

1.2 Python, Zope and Plone

Here I briefly explain what Python, Zope and Plone are.

Python is the foundational layer of the Zope and Plone combo. A few parts of Zope where raw processing speed is important C was chosen as the programming language. The rest is python. To paint a picture of what programming in python is like, I’ll simply quote the introduction from the official Python home page at http://python.org as I agree with that:
CHAPTER 1. INTRODUCTION

Python is a dynamic object-oriented programming language that can be used for many kinds of software development. It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days. Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.

The Zope Community site at http://zope.org says: “Zope is an open source application server for building content management systems, intranets, portals, and custom applications.” For building a website, plain simple Zope can be enough. At the time of this writing, my own home page at http://maurits.vanrees.org does that. You can install lots of custom products on Zope, for example an adapter to talk to your SQL database, a configurator for caching pages, authentication with LDAP, an interface to Google maps, etcetera.

The biggest products are probably the content management systems, like Silva, CPS and Plone. Parts of these systems rely on the CMF Content Management Framework that functions as a layer between them and Zope. The trend seems to be that the lower level CMF functions move into Zope. For eXtremeManagement it doesn’t really matter in practice whether a function that is called is officially part of Plone or CMF or Zope or maybe just a plain python function.

eXtremeManagement is built on Plone. Plone is a user-friendly and powerful open source Content Management System. Some of the highlights according to the plone.org website are that Plone is:

• easy to use: if you can operate a browser you can find your way through the Plone interface;
• easy to install;
• international (with more than fifty translation of the main interface);
• supported by hundreds of individuals and businesses;
• Open Source and that makes it extensible.

Plone is the layer that eXtremeManagement is built upon. The remainder of this report will therefore concentrate on Plone and not so much on Zope. Of course there will be a fair dose of python, as that is the glue that holds everything together.

1.3 eXtreme Programming

The eXtremeManagement product was made to help Zest in applying principles from Extreme Programming. That warrants an introduction.
CHAPTER 1. INTRODUCTION

Why is Zest Software using Extreme Programming? It keeps the programmers sane, the customer happy and Zest healthy and zesty. Extreme Programming allows you to plan just a few weeks ahead, deliver software to the customer and get paid for what you did. Rinse and repeat. Compare that to traditional development models that often go with fixed contracts and in the extreme case you get this: you build software for half a year, deliver it to the customer, he complains that you forgot several important points on page 3497 of the requirements document, which you will have to put in now at high costs and without extra pay as it was in the contract. Note that at this point you haven’t even been paid yet. Let’s hope that actually happens without a major court battle. For Zest Software it will not have been this bad, but fixed contracts were out. Extreme Programming showed promise to do a better job at keeping everyone happy and paid, so this development model was chosen.

Zest has been using the eXtreme Programming methodology with success for more than one year by:

- writing tests
- adhering to coding standards (XHTML, CSS2, WAGC, PEP-8)
- programming in pairs
- delivering in iterations of two to three weeks
- involving the customer in the development process.

Those are just some of the points of Extreme Programming. Ron Jeffries offers a high level overview [Jef]. For a more in-depth approach read the gentle introduction by Don Wells [Wel] or one of several books in the Extreme Programming series by Addison-Wesley. A main source of inspiration while making the eXtremeManagement tool was the book eXtreme Programming for Web Projects [WRA03].

1.4 eXtremeManagement

eXtremeManagement focuses on estimation and planning. Those are the central Extreme Programming activities that eXtremeManagement wants to help with.

Ahmed Hadi and Jean-Paul Ladage started working on the eXtremeManagement tool in February 2005. Its goal was to support our development team and to provide our customers with a unique way to monitor the progress on their project in real time. A lot of work was done, but there were too many rough edges to start using it yet. Some are listed in the next section.

My practice started at the end of November 2005. Its main goal was to make eXtremeManagement ready for use by Zest and ready for release to the general
public. Those goals were met. We have been enthusiastically using this tool in production since January 2006, and got a lot of positive feedback from our customers. The tool was also released as free software on plone.org as announced on February 27, 2006 as can be seen in appendix B on page 42.

1.5 Improvement possibilities

Some points that needed to be improved, were:

- the workflow
- get a list of booked hours
- send an email a task is assigned
- improve the input of hours in tasks
- add an estimate to the stories

A complete list can be seen in figure 1.1 on the following page. That is a screenshot from projects.zestsoftware.nl which is the internet site where Zest now keeps track of its projects using the eXtremeManagement tool. The page also lists the estimated and actual hours worked on each story. All stories have the status ‘completed’, which is good for an iteration that started on January 11 and ended on February 27.

January 11 was the day when the projects site was setup and that’s when Zest started using the tool in production. The time before that date was mostly spent learning a lot more about Plone and improving the workflows from eXtremeManagement. The time after the end of this first iteration was spent on several improvements for eXtremeManagement (see chapter 4 on page 27) and on work for actual clients (see appendix C on page 44).
Figure 1.1: Stories at the end of iteration 1.0

**Stories**

<table>
<thead>
<tr>
<th>story</th>
<th>status</th>
<th>estimate</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add our content types in the kupu_library_tool</td>
<td>completed</td>
<td>2:00</td>
<td>2:00</td>
</tr>
<tr>
<td>Start and End dates in Iterations</td>
<td>completed</td>
<td>3:00</td>
<td>2:30</td>
</tr>
<tr>
<td>notify employees about assigned tasks</td>
<td>completed</td>
<td>2:00</td>
<td>3:00</td>
</tr>
<tr>
<td>In task workflow: split up retract transition</td>
<td>completed</td>
<td>2:00</td>
<td>1:00</td>
</tr>
<tr>
<td>Add tests for Workflow</td>
<td>completed</td>
<td>6:00</td>
<td>1:15</td>
</tr>
<tr>
<td>CC the assigner of a task</td>
<td>completed</td>
<td>2:00</td>
<td>9:00</td>
</tr>
<tr>
<td>Send email on completion of task</td>
<td>completed</td>
<td>2:00</td>
<td>2:00</td>
</tr>
<tr>
<td>Fix manhours</td>
<td>completed</td>
<td>1:00</td>
<td>0:15</td>
</tr>
<tr>
<td>Allow discussion on Tasks and Stories by default</td>
<td>completed</td>
<td>1:00</td>
<td>0:30</td>
</tr>
<tr>
<td>Order stores as in navigation portlet</td>
<td>completed</td>
<td>1:00</td>
<td>1:00</td>
</tr>
<tr>
<td>Ability to order tasks</td>
<td>completed</td>
<td>0:00</td>
<td>0:00</td>
</tr>
<tr>
<td>Make portlets plone 2.1 compatible</td>
<td>completed</td>
<td>1:30</td>
<td>1:30</td>
</tr>
<tr>
<td>Get Bookings in a nice list</td>
<td>completed</td>
<td>7:00</td>
<td>12:00</td>
</tr>
<tr>
<td>Add a todo listing for employer</td>
<td>completed</td>
<td>6:00</td>
<td>6:00</td>
</tr>
<tr>
<td>Get a list of all your assigned tasks</td>
<td>completed</td>
<td>9:00</td>
<td>6:30</td>
</tr>
<tr>
<td>set bookingDate in update_hours_form</td>
<td>completed</td>
<td>2:30</td>
<td>2:30</td>
</tr>
<tr>
<td>Update workflow for Tasks</td>
<td>completed</td>
<td>6:00</td>
<td>10:15</td>
</tr>
<tr>
<td>Send an email when a Task is assigned</td>
<td>completed</td>
<td>5:00</td>
<td>2:15</td>
</tr>
<tr>
<td>Restrict editing permissions in several states</td>
<td>completed</td>
<td>2:00</td>
<td>2:15</td>
</tr>
<tr>
<td>Document the workflows</td>
<td>completed</td>
<td>6:00</td>
<td>2:00</td>
</tr>
<tr>
<td>Adapt workflow to avoid RST</td>
<td>completed</td>
<td>8:00</td>
<td>29:15</td>
</tr>
<tr>
<td>Make a release</td>
<td>completed</td>
<td>14:30</td>
<td>15:45</td>
</tr>
<tr>
<td>use hours and minutes for estimates</td>
<td>completed</td>
<td>31:00</td>
<td>22:15</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>122:30</strong></td>
<td><strong>135:00</strong></td>
</tr>
</tbody>
</table>
Chapter 2

Release 1.0: getting it ready

This chapter explains some of the bigger changes that have been made to the tool as it was at the end of November 2005 to get it ready for the first release. The chapter doesn’t focus on the changes per se, but more on the current state of the tool. Some content from the eXtremeManagement tutorial [LvR] is used. This tutorial was started by Jean-Paul Ladage and later brought up to date by me.

2.1 Workflows

The workflow from the tool was originally made via the Zope Management Interface through the web. It is more convenient to put this in the UML model (Unified Modeling Language). This makes it easier for others to understand the workflow. When changes are necessary, this is easier to do in the graphical UML model. Additionally, ArchGenXML can then automatically generate the needed code. So that’s what happened.

2.1.1 Content types

Before we focus on the workflow, an overview of the content types used can be enlightening.

ProjectFolder A ProjectFolder is a place for neatly storing all your Projects.

Project Multiple projects can be added by managers. For each project contact information of team members, iterations and stories can be added by both the customers and employees. You can also add a Poi Tracker for keeping track of bugs/issues.

Iteration The project will be planned with iterations. An iteration is a period of 1 to 3 weeks in which a number of stories will be implemented and integrated in a staging environment. Iterations contain stories.
**Story** The customer can define new features by describing this feature in a story. Stories contain tasks.

**Task** The employees can estimate the story by defining tasks. Tasks contain bookings.

**Booking** After working on a task, the employee can book his hours.

### 2.1.2 Roles

On these content types, different roles have different permissions. So which roles are there again?

**Manager** Only managers are allowed to add and modify projects. They are also the only ones who can put an Iteration on invoiced, signalling that a bill has been sent to the customer.

**Employee** Employees can add and modify Iterations, Stories and Tasks. You usually want to give each of your developers this role globally so they can work on each project. If your development team is large, you may want to give these only as local roles per project. For assigning local roles, use the project team tab on your project.

**Customer** The customer can only add stories and submit them for estimating and manage the priority of the stories by assigning them to iterations. This should be a local role given only for that customer’s project.

The combination of workflow and roles can be used to keep development on track. For instance when a Story is completed, it would be strange to add a Task to it. In some cases you may want this after all, but a normal Employee can’t do that. Only a Manager can help with this if needed by changing the state of the Story. That keeps the manager in control and prevents the employee from doing things that the customer didn’t request.

### 2.1.3 Project workflow

Projects start out in the private state. A Manager (or Owner, but that’s usually the same person) can activate a project. Once the project is active, the customer can be involved. When all is said and done and the Manager/Owner can also close the project. Those two transitions are shown in blue, the colour we use in all diagrams to indicate that this is the normal route. If you made a transition and later realize you made a mistake, you can go back to the previous state. This is true for most of the transitions in the other workflows as well.

When you add this as a state diagram to your Unified Modeling Language model and add some tagged values, ArchGenXML can generate the necessary
code for you. To get a feel for what the result is, here follow some lines from the python code for the project workflow.

First the states, transitions and some standard workflow variables are added:

```
for s in ['active', 'completed', 'private']:
    workflow.states.addState(s)

for t in ['deactivate', 'close', 'activate', '← reactivate']:
    workflow.transitions.addTransition(t)

for v in ['review_history', 'comments', 'time', '← actor', 'action']:
    workflow.variables.addVariable(v)
```

Some permissions are added:

```
workflow.addManagedPermission('Access contents information')
workflow.addManagedPermission('Add portal content')
# There are more here
```

The initial state is set to 'private':

```
workflow.states.setInitialState('private')
```

Then the states are initialised by giving them names, appointing transitions to other states and configuring permissions for certain roles:

```
stateDef = workflow.states['active']
stateDef.setProperties(title=""Active"",
    transitions=[['close', '← deactivate']])
```
2.1.4 Iteration workflow

See the state diagram in figure 2.2.

After adding an iteration, you can start adding stories to the iteration and tasks to the stories. At a certain moment you aught to have an iteration start meeting with the customer, finalising the set of stories. Afterwards, you can set the workflow to ‘in progress’, which activates the stories and the tasks. The bright yellow note (grey actually when you look at a black and white copy of this report) that grabs your attention indicates a transition that is triggered by another transition. In this case: if all stories belonging to this iteration have been completed, the iteration itself is set to completed. After this, the Manager can invoice the iteration, which indicates that a bill has been sent to the customer.
2.1.5 Story workflow

See the state diagram in figure 2.3.

The customer adds a story, describes it and finally submits it for estimation. The manager/developer then provides a rough estimate (in days). Then the story can be set to the ‘estimated’ state. The developers can now add tasks.

The estimated story is activated automatically when the iteration is activated. This activation also triggers the activation of the tasks. The iteration should be only activated when all stories have tasks and all tasks are estimated (an estimate greater than 0) and have at least one assignee. The workflow protects you from activating when there are unestimated tasks.

If all tasks have been completed, the story is automatically also completed. If all stories have been completed, the iteration is automatically also completed.
2.1.6 Task workflow

See the state diagram in figure 2.4.

Tasks are added by developers. Like mentioned before, when all tasks are estimated, their parent story becomes startable. When all stories are startable, the iteration can be started. So when you view an iteration and you are allowed to start it, this means that all its stories and all their tasks are also startable. When you then actually click on ‘start’ then indeed this iteration and all its stories and tasks are activated.

If all tasks in a story are completed, the story is automatically marked as completed.

Note: Tasks used to have more states. They were removed when the eXtreme-Management tool was already in use. That meant that some migration had to be done. Tasks with a state of ‘in-progress’ needed to get a new state ‘to-do’; Tasks with a state of ‘estimated’ or ‘assigned’ needed to get a new state ‘open’. To make this work involves all kinds of trickery; at least it seems that way when you have never done it before. For details look at the subversion repository ¹.

2.1.7 Booking workflow

The workflow for Bookings is actually not worth the term ‘workflow’. It has only one state, simply called ‘booking’. Let’s instead show some code that can be run automatically to test if this is true.

First a helper function:

¹https://svn.zestsoftware.nl/svn/zest/eXtremeManagement/trunk/Extensions/MigrateWorkflow.py
def tryForbiddenTransition(self, ctObject, originalState, workflowTransition):
    """
    Try to execute a transaction that you are not allowed to do
    ctObject = ContentType object to perform the transition on
    originalState = current state of the object
    workflowTransition = transition to perform
    """
    self.assertEqual(self.workflow.getInfoFor(ctObject, 'review_state'), originalState)
    self.assertRaises(WorkflowException, self.workflow.doActionFor, ctObject, workflowTransition)

This asserts that the original state of an object is what we expect and then
asserts that trying to do a certain transition raises an exception.

Now give the current user the role 'Manager' and try two non-existing transitions:

def test_booking_transitions(self):
    """
    Test transitions of the Booking ContentType.
    Hm, there aren't any transitions here.
    """
    self.setRoles(['Manager'])
    self.tryForbiddenTransition(self.booking, 'activate')
    self.tryForbiddenTransition(self.booking, 'submit')

The test then runs without errors, showing that the assertions mentioned above were right.

2.2 Task estimates

Tasks used to have an estimate that contained the number of hours that the task
is expected to take. Only whole hours could be entered though. We wanted to
be able to specify hours and minutes.

This works now. See figure 2.5 on the following page. This is actually not that
hard using Archetypes and ArchGenXML. In your UML model you just rename
CHAPTER 2. RELEASE 1.0: GETTING IT READY

Figure 2.5: Task estimate

Figure 2.6: Task class
the ‘estimate’ variable to ‘hours’ and you add a variable called ‘minutes’. The
Task class now looks like figure 2.6 on the page before. Now add some tagged
values to the new variable ‘minutes’, like in figure 2.7. Run ArchGenXML and
you’re done!

Except that some things will now go wrong. New Tasks will go fine. But Tasks
that are currently in your Zope instance will still have that variable ‘estimate’
instead of ‘hours’ and will not have ‘minutes’. Also, when you have not yet
changed the view template (html page) for tasks, then viewing new tasks will
give an error message, as the template will want to display the estimate of the
Task, but can’t find it. When you repair that by letting it display the hours and
minutes instead, then old Tasks can’t be viewed anymore, because they won’t
have those fields.

The problems for the template can be worked around by putting some con-
ditions in it. But it is really unwise to have versions of a content type with a
different schema in your instance. So what you do is you update the Task schema.
You can do that by hand or by calling this line in your reinstall script:

```python
_migrateTaskSchema(portal)
```

But this is not enough. This just adds 0 hours and 0 minutes to each Task
and might not even throw away the estimate. So you need to be smart. First get
a list of all current Tasks:

```python
task_brains = portal.portal_catalog(meta_type='Task')
print >> out, "Found %s tasks." % len(task_brains)
```

A brain is actually a small portion of a Task (or any other content type),
namely just the part that is available via the catalog, which is very quick. Then
build a list of tasks that still have an estimate. In that list put the task brain
and its current estimate:

```python
tasklist = []
for task_brain in task_brains:
    task = task_brain.getObject()
    if hasattr(task, 'estimate'):
        old_estimate = getattr(task, 'estimate')
```

Figure 2.7: Tagged values of Task
tasklist.append((task_brain, old_estimate))

Now migrate the schema of the Tasks:

_migrateTaskSchema(portal)
print >> out, "Number of tasks that need to be migrated = %s." % len(tasklist)

Now go through the list of old tasks again and set their `hours` to the old value of their estimate and to be safe throw away the estimate:

for item in tasklist:
    task_brain, old_estimate = item
    task = task_brain.getObject()
    task.setHours(old_estimate)
    print >> out, "Migrating task %s with estimate of %s hours." % (task.title, old_estimate)
    if hasattr(task, 'estimate'):
        delattr(task, 'estimate')
    task._updateCatalog(portal)

print >> out, "Migration of tasks completed."

And your migration is done.

2.3 Overview of bookings

One of the things missing when I started with eXtremeManagement was an overview of your bookings. How much did you work today? This month? And what exactly did you do?

Figure 2.8 on the next page shows a portlet called ‘Project Administration’ that helps with this. It appears on the right hand side of all pages now. It contains some links to handy pages, like the booking overviews that will be further introduced below. It also shows the hours you booked today, which is useful to see at a glance.

Clicking on “Bookings per month” shows you an overview of your total hours booked per day in a month, like in figure 2.9 on page 17. When you want to go into more detail, you can go to “Detailed bookings”, which looks like figure 2.10 on page 18. In this case these pages give info on every project, but you can also call them on just one project, or one iteration, etcetera.

And anyway, you can see all bookings on a task simply by viewing that task. For the task of writing this section on booking overviews, the status so far can be seen in figure 2.11 on page 19, taken from the Zope instance on my local computer. By the way, this shows that eXtremeManagement can also be used on
personal projects like making a report. It is not exclusively a tool for groups of developers.

Finally, figure 2.12 on page 20 has some code showing the basis of the calculations that are done in the background for the overview pages.
Back to current month.

Bookings per day

<table>
<thead>
<tr>
<th>date</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-04-03</td>
<td>9:30</td>
</tr>
<tr>
<td>2006-04-04</td>
<td>6:15</td>
</tr>
<tr>
<td>2006-04-05</td>
<td>7:30</td>
</tr>
<tr>
<td>2006-04-06</td>
<td>8:15</td>
</tr>
<tr>
<td>2006-04-10</td>
<td>7:30</td>
</tr>
<tr>
<td>2006-04-11</td>
<td>8:45</td>
</tr>
<tr>
<td>2006-04-12</td>
<td>8:15</td>
</tr>
<tr>
<td>2006-04-13</td>
<td>8:00</td>
</tr>
<tr>
<td>2006-04-18</td>
<td>8:00</td>
</tr>
<tr>
<td>2006-04-19</td>
<td>9:00</td>
</tr>
<tr>
<td>2006-04-20</td>
<td>7:45</td>
</tr>
<tr>
<td>2006-04-24</td>
<td>8:00</td>
</tr>
<tr>
<td>2006-04-25</td>
<td>7:30</td>
</tr>
<tr>
<td>2006-04-26</td>
<td>6:45</td>
</tr>
<tr>
<td>2006-04-27</td>
<td>7:45</td>
</tr>
<tr>
<td>2006-04-28</td>
<td>1:30</td>
</tr>
</tbody>
</table>

Figure 2.9: My bookings for the month April
Go to previous month.

**Bookings**

<table>
<thead>
<tr>
<th>date</th>
<th>task</th>
<th>booking</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006-05-01</td>
<td>Harm Bouwman toegang geven tot subversion</td>
<td>Harm Bouwman toegang geven tot subversion</td>
<td>0:30</td>
</tr>
<tr>
<td>2006-05-02</td>
<td>meetings</td>
<td>Overleg met Frank</td>
<td>0:30</td>
</tr>
</tbody>
</table>
| 2006-05-02 | Issue 33: Ploneboard              | Fixed
Uploaded change to Plone collective. Uploaded fresh tar.gz in our Products. | 1:00  |
| 2006-05-02 | Issue 25: Searching word documents | More investigation
Last week it worked locally but not on the server. Now it also doesn't work locally. Sigh :( | 1:00  |
| 2006-05-02 | Issue 30: Recent changes vs. Modified items | Fixed | 0:15  |
|            | Issue 29: spell checker            | fixed
Copied a javascript file from fckeditor to philispayout and changed the version string there. Will happen again when the version we now link to is removed from that download site, though. | 2:15  |
|            | Fix bug 20: changing ownership    | Tried new code for checking if user exists
Not tested yet. | 0:45  |
|            | Issue 32                          | Made sure that you can reach the file via the contents tab of the parent. | 0:30  |

Figure 2.10: My booking details for the month May
### Figure 2.11: Bookings for the task of writing this section

<table>
<thead>
<tr>
<th>Date</th>
<th>Booking</th>
<th>Time</th>
<th>Booked by</th>
<th>Billable</th>
</tr>
</thead>
<tbody>
<tr>
<td>2006/05/04</td>
<td>Investigating why this month doesn't show bookings</td>
<td>0:45</td>
<td>maurits</td>
<td>True</td>
</tr>
<tr>
<td></td>
<td>Did a catalog update, but that didn't help. Will this booking show up?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2006/05/04</td>
<td>Added screenshots plus bug fixed in xm</td>
<td>1:15</td>
<td>maurits</td>
<td>True</td>
</tr>
</tbody>
</table>
# Get today's date
date = DateTime()

# Get 00:00:00 of that date
startDate = DateTime.earliestTime(date)

# Get 23:59:59 of that date
endDate = DateTime.latestTime(date)

# Get a list of all Bookings in the context which
# have a booking date within the range of startDate
# and endDate, in other words: today.
bookingbrains = context.portal_catalog.searchResults(
    portal_type='Booking',
    getBookingDate={ "query": [startDate, endDate], "range": "minmax"},
    Creator=memberid,
    path=searchpath)

# Calculate formatted total (e.g. 3:15) as the sum of
# all rawActualHours booked.
total = 0
if bookingbrains:
    actualList = []
    for bb in bookingbrains:
        booking = bb.getObject()
        actualList.append(booking.getRawActualHours())
    total = sum(actualList)
    total = formatTime(total)

return total

Figure 2.12: Python script getDailyBookings.py
Chapter 3

Releasing a Plone product

We released eXtremeManagement on plone.org. What steps did we take to get it available there? I generalised that process and put it in a tutorial on the documentation section of that website. This chapter is that tutorial [vR].

3.1 Introduction

After months of hard labour you have finally finished your shiny new Plone product. You are ready to receive fan mail from happy users. So now what should you do?

3.2 Product versions

It is important to know which version of your product is installed. This especially helps with bug reports. Is someone using 1.0 from your tar ball? A bug fix release from a branch? The cutting edge version from trunk? Read on for some advice on version strings and source code control with subversion.

3.2.1 Version strings

You have just finished ShinyNewProduct and think it is ready to take the market by storm. Great! Let’s release it then. Of course you have developed your product in a source code control system. The favourite among Plone developers seems to be subversion, or svn for short. This tutorial will focus on that, but the concepts should be applicable to CVS (Concurrent Versioning System) or other systems. There are also tips on using CVS instead of subversion [Bry].

First of all, your product needs to have a version.txt. You can argue a lot about the numbers that need to go in there, but your ShinyNewProduct feels like a 1.0 release, so let’s just put that in there and commit the change, which goes something like this:
echo "1.0" > version.txt
svn commit -m 'Prepare release 1.0'

### 3.2.2 Branches, tags and trunk

Let's say you have your product in http://localhost/svn/ShinyNewProduct. Your actual code should be in a directory `trunk`. If you don’t have that, it is beyond the scope of this tutorial to help you change that. What I will do is show you how to add two more directories. If you already have them, great!

```
svn mkdir http://localhost/svn/ShinyNewProduct/branches
svn mkdir http://localhost/svn/ShinyNewProduct/tags
```

The main development is done on trunk. You could choose to develop only on branches instead. In fact, for big changes that take longer than a day to finish, this should be standard practice. It is easy to copy the trunk to a branch, develop code there, merge the changes back to trunk when you’re happy and possibly remove the branch again.

Anyway, assume for now that your latest greatest code is in trunk. Now, when people start using your product, they will probably find bugs. That’s okay, bugs thrive on software, so it’s to be expected. There’s no need to be ashamed of yourself. Just fix the bugs. But your users will not only find bugs, they will also come up with feature requests; and if not, then you yourself will. You made the decision to develop new features on trunk. But when you fix a bug your users want to be able to use version 1.0 plus the bug fix, without any new features which may be buggier still (or so they think). So then where do the bug fixes go?

Bugs will need to be fixed in your latest code and in the code that you just released as version 1.0. So at release time you copy your trunk to a branch:

```
```

When bugs surface you can fix them on that branch and merge the changes back to trunk.

Sometimes you want to be able to go back to the exact version of your product at release time. Your clever bug fix may have broken something so this branch is unusable at the moment. That’s when the `tags` subdirectory comes into play. At release time you should do

```
svn copy http://localhost/svn/ShinyNewProduct/branches/1.0 \http://localhost/svn/ShinyNewProduct/tags/1.0
```
CHAPTER 3. RELEASING A PLONE PRODUCT

Whether you copy trunk to branches to tags or trunk to tags to branches shouldn’t really matter, as long as you don’t commit anything in the mean time. Note that committing changes to a tag is a violation of best practices. Anyone should be able to count on the fact that tags/1.0 really contains the code as it was when version 1.0 was released, without any bug fixes or other changes. There are ways to actually prevent committing to an existing tag, but that’s beyond the scope of this tutorial.

Now is a good time to change the version.txt in branches and trunk to signal the new state of the software. After the first release, the version.txt should look as follows:

In tags/1.0: 1.0
In branches/1.0: 1.0 svn/maintenance build 0
In trunk: 1.1 svn/devel build 0

After you commit changes to the branch or trunk you can increase the version.txt to signal the change (build 1, build 2, etcetera). Note that we have given trunk a version number starting with 1.1. You do want to continue developing don’t you?

3.2.3 Make a tar file for downloading

Now you can prepare a tar file that people can download from a website when they don’t want to use subversion. Best is to make an export from the tag. An export instead of a checkout ensures that there are no subversion directories or old .pyc or backup files lying around. Double check to make sure the version number in version.txt is plainly 1.0.

```
svn export http://localhost/svn/ShinyNewProduct/tags/1.0 \
ShinyNewProduct
tar czf ShinyNewProduct-1.0.tar.gz ShinyNewProduct
```

Okay, you now have a trunk, tags and branches with version numbers to match and you have a tar ball. You have taken a big step towards releasing your ShinyNewProduct. Actually, the most difficult part is finished now, so you can breathe and relax. Then again, you still need to tell your future users what your product does, where they can download it and how they can contact you to express their gratitude (or perhaps how to complain, but that’s a corner case). Read on for more advice on that.

3.3 Steps on plone.org

What should you do to get your product on plone.org?
3.3.1 Plone Software Center

First of all, login to plone.org. What? You don’t have an account there yet???
Quick, go subscribe immediately before someone notices. For the others: go on a coffee hunt; you deserve it!

Ah, that’s better. Right, you are logged in. Now go to plone.org/products.
You should be able to register a new product there by clicking the button marked “Add new software project”. This will show you an edit form where you can add a new Plone Software Center. This comes with a tutorial [Bur] that you really want to read as it has lots of tips that I fail to mention here.

Basically, here you can point to a home page, a contact address, a code repository, etcetera. Everything you always wanted. Add the information that is relevant for your project. Maybe add some documentation already or write some improvement proposals or add a bug tracker. Then submit your project for review and wait patiently for one of the reviewers to make your project public.

3.3.2 Place of code repository

You can keep your subversion repository on a server that you yourself maintain. If your product is any good, you will sooner or later get patches from other developers. If they are really enthusiastic, they will even ask for commit access to your repository as they want to submit changes themselves, perhaps on a branch. That is certainly a workable solution, but you may want to spare yourself some hassle and follow the pack to the plone subversion repository. Probably the collective is the best spot, at https://svn.plone.org/svn/collective/. There is a how-to on getting write access to the plone collective svn repository [Sta].

Now what am I forgetting? Ah, remember that tar file you made earlier? Add a release to your product page and add the tar file there so people can download it. When you are ready publish the release and prepare for an onslaught of users. But wait! Don’t do that yet. You may first want to setup some mailing lists so people can ask questions and search for answers. Read on.

3.4 Mailing lists

You may want to setup one or two mailing lists specifically for your product. Having those available as newsgroups on gmane.org is nice as well.

3.4.1 User mailing list

Don’t look here for advice on which mailing list software to use. You may want to use some ready made lists like on sourceforge.net. You can also decide to just point users to the general plone-users mailing list. That’s fine. See http://plone.org/support.
CHAPTER 3. RELEASING A PLONE PRODUCT

But if someone has a problem with your product and mails the plone-users list with a subject line of just ‘Problem’ you are likely to ignore that message. If he instead uses a mailing list specifically aimed at your product, you are much more likely to read his mail. Well, that’s one reason to create a mailing list for users.

If your product is very successful and also draws attention from a lot of developers who want to embrace and extend your code, you could add a mailing list specifically for discussing development. But stick to one user list for now.

3.4.2 Issue mailing list

There is one other list that you may want to add: a list where bug reports are automatically sent. For example if you added a Poi issue tracker to your product page in an earlier step, you can choose to send any bug reports to a few email addresses or you could let them be sent to a mailing list. Anyone interested can subscribe to that list. This safe you some hassle.

3.4.3 gmane newsgroup

When setting up the user list for my own eXtremeManagement product, some fellow developers told me they would like the list to be available as a newsgroup via gmane.org. That is fairly easy to setup, so I suggest you do that as well. Just visit http://gmane.org/add.php and follow instructions there. I’ll add one remark: to make sure users can easily find the newsgroup, make sure you stick to the naming convention for plone lists. In this case I would suggest as a group name: gmane.comp.web.zope.plone.shinynewproduct.

After some time (could be several days) you normally get an email with a courtesy copy of an article that has been posted to the gmane.discuss.subscribe newsgroup as well. Follow any instructions that you read there, if any. Usually this will tell you that everything is ready and the newsgroup will automatically be created after the next email is sent to the mailing list.

Make sure you mention your new mailing lists and newsgroups on your products page in the Plone Software Center on plone.org. Okay, now you can start using the lists. How about announcing the first release of your product there? Read on.

3.5 Announcement

You want potential users to notice your new product. Where should you send the announcement? What should be in the announcement?
3.5.1 Content of announcement

“Please download my eXtremeManagement product. It is very good. Click here: http://plone.org/products/extreme-management-tool!”

Did you just click that link? Yes? Then spammers must love you. No? Okay, then you agree that your announcement should be better than that. For starters, you want to provide more information, to entice your readers to follow that link. Your potential users will have some questions:

- What is the name of your product?
- Who is releasing this? Is it a product just from you or does it have a company behind it?
- What does it do?
- What does it do differently than other similar products?
- Is it a Zope, CMF or Plone product?
- On which Zope and Plone versions has it been tested and known to work?
- What other products does it depend on?
- Where can I download it?
- Where can I get more info?

Don’t overwhelm your readers though. Two screens full of information should be the limit. If you need more than that, provide some links instead. Just concentrate on clearly saying what your product does and throw in a link to more detailed information, like install instructions.

3.5.2 Where to send it

Are you done writing? Great! Have you done a spell check? Wonderful! Has it been proofread by co-workers or so? Even better! Now send it out. If you made a mailing list, send it there first. Then inform the plone community. Go to http://plone.org/support and find the address of the plone-users mailing list or newsgroup and post it there. Don’t use the plone-announcement list: that is only for core Plone itself, not for Products. Are you often found at irc? Then throw a line or two in the #plone channel.

Now let the feeling sink in that you have really released a Plone product. Congratulations. Now throw a party and do the happy dance: http://www.schlockmercenary.com/d/20040822.html!

\(^1\)For the real announcement message of the eXtremeManagement tool see appendix B on page 42
Chapter 4

Release 1.1: new features proposed

Several new features have been proposed by colleagues, people from around the Internet and myself. Figure 4 on the next page shows the stories that have been proposed for iteration 1.1, the iteration that will lead to release 1.1. We haven’t yet decided which stories will indeed be implemented and which will be deferred to the next iteration.

Section 4.1 explains a feature that has already been completed: auto-assigning new Tasks to the writer.

Section 4.2 on page 31 discusses ways to better indicate how to signal the progress of a Task: if a task is estimated at 2 hours and you have already worked 3 hours on it, but aren’t finished, then what is the progress percentage? 150 percent would be strange.

Section 4.3 on page 34 brings forward some thoughts on integrating Martin Aspeli’s Poi issue tracker with eXtremeManagement.

Section 4.4 on page 35 speaks about moving the eXtremeManagement code to the Plone collective repository.

4.1 Auto-assign new tasks

Let’s look at one new feature that has already been implemented. When you add a new task, that tasks is automatically assigned to yourself. The edit form that you see when you add the task will have your name selected as an assignee. You can of course choose someone else at that time, but quite often you will assign a task to yourself, so it is a sensible default.

In principle this is not so hard. The Task attribute ‘assignees’ has already been defined in the Archetype Schema and you only need to add a line pointing to a default_method:

LinesField(
### Stories

<table>
<thead>
<tr>
<th>story</th>
<th>status</th>
<th>estimate</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto-assigning of new tasks to yourself</td>
<td>completed</td>
<td>4:00</td>
<td>5:30</td>
</tr>
<tr>
<td>Set the correct order for Iterations, Stories, Tasks</td>
<td>completed</td>
<td>1:30</td>
<td>2:45</td>
</tr>
<tr>
<td>Switch to Poseidon 4.0</td>
<td>completed</td>
<td>1:00</td>
<td>0:30</td>
</tr>
<tr>
<td>Install Plone 2.1.2 on server</td>
<td>completed</td>
<td>5:00</td>
<td>12:45</td>
</tr>
<tr>
<td>Install FCKEditor</td>
<td>completed</td>
<td>2:00</td>
<td>1:00</td>
</tr>
<tr>
<td>Dates of bookings</td>
<td>estimated</td>
<td>2:00</td>
<td>1:30</td>
</tr>
<tr>
<td>Better progress indicators</td>
<td>estimated</td>
<td>2:00</td>
<td>2:00</td>
</tr>
<tr>
<td>Give Managers a better insight in Bookings</td>
<td>draft</td>
<td>8:00</td>
<td>0:00</td>
</tr>
<tr>
<td>Improve Layout</td>
<td>estimated</td>
<td>4:00</td>
<td>0:00</td>
</tr>
<tr>
<td>Add worklists</td>
<td>draft</td>
<td>4:00</td>
<td>0:00</td>
</tr>
<tr>
<td>enhancements to story_view.pt</td>
<td>draft</td>
<td>0:00</td>
<td>0:00</td>
</tr>
<tr>
<td>After adding a task redirect to the parent story</td>
<td>estimated</td>
<td>16:00</td>
<td>0:00</td>
</tr>
<tr>
<td>A story should be assigned to one accountable member</td>
<td>estimated</td>
<td>12:00</td>
<td>0:00</td>
</tr>
<tr>
<td>Detailed bookings should also be viewable per month</td>
<td>estimated</td>
<td>1:30</td>
<td>1:45</td>
</tr>
<tr>
<td>Add translations</td>
<td>estimated</td>
<td>1:30</td>
<td>0:45</td>
</tr>
<tr>
<td>Small bug fixes</td>
<td>estimated</td>
<td>3:45</td>
<td>2:45</td>
</tr>
<tr>
<td>Small unintrusive changes</td>
<td>estimated</td>
<td>4:30</td>
<td>5:15</td>
</tr>
<tr>
<td>Think and talk about xm</td>
<td>estimated</td>
<td>4:00</td>
<td>0:45</td>
</tr>
<tr>
<td>Emails on tasks</td>
<td>estimated</td>
<td>2:00</td>
<td>0:45</td>
</tr>
<tr>
<td>Add documentation</td>
<td>estimated</td>
<td>9:00</td>
<td>0:00</td>
</tr>
<tr>
<td>Move repository to collective on plone.org</td>
<td>estimated</td>
<td>8:00</td>
<td>0:00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>95:45</strong></td>
<td><strong>38:00</strong></td>
</tr>
</tbody>
</table>

Figure 4.1: Stories for iteration 1.1
name='assignees',
default_method='getDefaultAssignee',
index='FieldIndex'",
widget=MultiSelectionWidget (  
description="Select the member(s) to assign this task to.",
label='Assignees',
label_msgid='eXtremeManagement_label_assignees',
description_msgid='eXtremeManagement_help_assignees',
i18n_domain='eXtremeManagement',
),
multiValued=1,
vocabulary='_get_assignees'
),

Now we need to define that method:

def getDefaultAssignee(self):
    mem = getToolByName(self, 'portal_membership')
currentUser = mem.getAuthenticatedMember().getId()

    if currentUser in self._get_assignees():
        return currentUser
    else:
      
        return ''

This calls the function _get_assignees which returns a list of team members. By the way, Jodok Batlogg and others from Lovely Systems ¹ have jumped on the eXtremeManagement wagon and are busy rewriting this part so it can use LDAP and other authentication methods.

Everything seems to work. So far, so good. But we changed the Schema of Task, so the Schema of the current Tasks needs to be updated, like we did in section 2.2 on page 12. A local test of this on the development machine did just fine, so now we update the instance on the Zest projects server. Reinstall takes very long, suspiciously long. Then a colleague mentions that he is getting a lot of emails from eXtremeManagement with tasks that are assigned to him. Now, eXtremeManagement will send you an email when someone else assigns a Task to you, but in this case it was simply spamming everyone. But why?

It turns out that during a schema update for every old Task \( T \) a new Task \( U \) is created. Each attribute from \( T \) is then copied to \( U \), if that attribute still exists. The part of our code that determines if an email needs to be send, sees a new

¹http://www.lovelysystems.com/
assignee and sees that this is not the current user, so it sends out an email. Add
a few hundred tasks and you can imagine that the update takes quite long and
spams everyone with emails. Not good.

Armed with this info, first the part of the code that does the schema migration
got smarter:

```python
def _migrateTaskSchema(self):
    
    Add a property to the portal so that other parts
know that there is a schema update going on for
the Tasks. Main reason: if this is True, then
_do not_ send an email for every Task that is
getting assigned. See Task.setAssigned()

    
    propertyName = 'xm_task_schema_updating'
    self.manage_addProperty(propertyName, True, 
    boolean)
    _migrateSchema(self, 'eXtremeManagement.Task')
    self.manage_delProperties((propertyName,))
```

The `setAssignees` method got improved with lots more checks, including the
check for the `xm_task_schema_updating` property that just got introduced. Here
is the result:

```python
security.declarePublic('setAssignees')
def setAssignees(self, value, **kw):
    
    Overwrite the default setter. An email should
be sent on assignment.

    But not when the Task is edited and the assignees
don’t change. And if they _do_ change, then don’t
mail the people that were already assigned.

    Now why does setAssignees get called *three*
times when a new Task is made???

    And why is the value a list which contains an
empty item ' '???

    Anyway, we need to do some serious checking.

    if isinstance(value, StringTypes):
        value = [value]
```
self.log.debug('New assignees value=%s.', value)
old_assignees = list(self.getAssignees())
self.schema['assignees'].set(self, value)
while '' in value:
    value.remove('')
if old_assignees != value:
    self.log.debug('old_assignees=%s.', old_assignees)
portal = getToolByName(self, 'portal_url').getPortalObject()
if portal.hasProperty('xm_task_schema_updating'):
    self.log.debug('Task schema update, so not sending email to %s for task %s.',
                    value, self.id)
else:
    for employee in value:
        if employee not in old_assignees:
            self.log.debug('Sending email to %s for task %s.', employee, self.id)
            mailMessage(portal, self, 'New Task assigned', employee, self.log)
        else:
            self.log.debug('Not sending email: %s was already assigned to task %s.',
                           employee, self.id)

4.2 Better progress indicators

In the project view you can see an overview of how far the stories have progressed in the current iteration. Figure 4.2 on the next page shows such a page for the project of creating this report. This screen shot actually already has some improvements compared to the situation that Reinout talks about in the eXtreme-Management issue tracker ²:

The current progress indicator is really a (hours worked)/(hours estimated)*100 percent indicator. That's an OK view, but I want another one that tries to be a bit more intelligent about it.

²http://plone.org/products/extreme-management-tool/issues/4
### Figure 4.2: Progress indicators

<table>
<thead>
<tr>
<th>story</th>
<th>progress</th>
<th>tasks open</th>
<th>tasks completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 1: Introduction</td>
<td>100 %</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2: Release 1.0: getting it ready</td>
<td>100 %</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 3: Releasing a Plone product</td>
<td>100 %</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Add screenshots</td>
<td>0.0 %</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Design front page</td>
<td>0.0 %</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Preface</td>
<td>0.0 %</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Abstract</td>
<td>0.0 %</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Chapter 4: Release 1.1: new features</td>
<td>29.4 %</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Conclusion</td>
<td>0.0 %</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Bibliography</td>
<td>0.0 %</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Appendices</td>
<td>18.8 %</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
So: if a completed task only took 1 hour versus a 4 hour initial estimate, count it as 4 hours instead of the 1 hour that’s used now.

Idea is also that the current progress shouldn’t go over 100 percent. You can’t really be more-than-100 percent-finished. So tasks that aren’t finished yet but are nearing their estimated duration should be (gradually) cut off. Especially important for task that take longer.

This should give the customer (and the developer) a much more reliable (though perhaps a bit conservative) estimate of the progress.

Some of this is already handled differently now. When the status of a Story is set to ‘completed’, the progress percentage is reported as 100 percent, no matter what the original estimate and the hours booked were.

Also, progress percentages can no longer go over 100 percent. A Story that has already taken for instance two hours instead of one will not have a progress percentage of 200. Instead, the parameter MAXIMUM NOT COMPLETED _PERCENTAGE of 90 was introduced as a cutoff percentage. Any progress above that will not be shown, so as not to give a false impression of the Story being almost completed. This is best shown with some tests:

```python
self.assertEqual(MAXIMUM_NOT_COMPLETED_PERCENTAGE, 90)

# Estimate that self.task will take 1 hour:
self.task.setHours(1)
self.assertEqual(self.story.get_progress_perc(), 0)

# Book 15 minutes on the task:
self.task.invokeFactory('Booking', id='booking1', hours=0, minutes=15)

# This should result in the parent story having a # progress of 25 percent:
self.assertEqual(self.story.getRawActualHours(), 0.25)
self.assertEqual(self.story.get_progress_perc(), 25)
self.task.invokeFactory('Booking', id='booking2', hours=0, minutes=45)
self.assertEqual(self.story.getRawActualHours(), 1.0)

# 1 of 1 estimated hours has been booked.
# The Story is not completed yet, so the progress # should be MAXIMUM NOT COMPLETED _PERCENTAGE (90) # percent:
self.assertEqual(self.story.get_progress_perc(), 90)
sself.login('employee')
```
self.task.setAssignees('employee')
self.workflow.doActionFor(self.story, 'activate')
self.workflow.doActionFor(self.task, 'complete')

# Now the Story is completed, so progress is 100:
self.assertEqual(self.story.get_progress_perc(), 100)

Now, this is hardly ideal. Various proposals have been done: add red and green colours to signal overtime and ‘undertime’; give tasks or stories an original and a current estimate. The discussion at the Zest office continues. We’ll think of a good way to solve this.

4.3 Integration with the Poi issue tracker

Another issue that needs some more debating is integration with Poi\(^3\). Poi is a simple issue (bug) tracker for Plone. At the Zest projects site most of our projects by now have a Poi tracker inside their eXtremeManagement Project-Folder. Customers can file bug reports and other issues there. It’s becoming standard practice to create a Story called something like “Bugs and UI changes from the bug tracker”. Inside that story you can then create a Task for every issue that you work on. This feels a bit like double work, so Reinout started a discussion about this\(^4\):

We have Poi issue trackers and XM tasks. When fixing a bug, we want one task per bug. But creating a task, copy/pasting an issue URL and so: that’s a lot of work.

This could be mostly automated by adding a Task-subclass called IssueTask with the build-in capability to link to a Poi issue. The title should be set from the issue name (preferably something like “issue 23: name of issue”). That way you’d only have to select the corresponding bug and you’d be set. You only need to add bookings to this task, that’s all.

Something like this would indeed be good to have. This might be a nice opportunity to try out some of the possibilities of Five/Zope 3. We could then do this cleanly with interfaces. Both the current Task and the new IssueTask would implement an ITask interface. This then opens possibilities for integration with other tracker products.

The IssueTask still needs an estimate and an assignee. The assignee could probably be read from the Poi issue. If no responsible manager is known there

\(^3\)http://plone.org/products/poi
\(^4\)http://plone.org/products/extreme-management-tool/issues/13
the IssueTask could try setting the Task assignee and the issue manager in one go.

We could arrange that a state change in IssueTask results in a state change in the Poi issue. On creation of a new IssueTask the Poi issue state could be set to 'open'. When the IssueTask is set to 'to-do' the Poi issue could be set to 'start working' or whatever it is called. When the IssueTask is completed, it could get trickier, as this probably means that the Poi issue can be set to 'resolved', but the issue might also have been postponed or rejected. Automatic state changing can wait though.

4.4 Moving to the collective

The code of eXtremeManagement is now housed on the subversion repository of Zest Software. According to section 3.3.2 on page 24 this isn’t the best place. It should instead be put in the plone collective. Since the end of April I have write access to the collective, so putting it there is certainly an option.

How would we do that? First of all, Lovely Systems is working on a branch of eXtremeManagement. Before moving to the collective we want that branch merged with trunk. We’ve had contact with them this morning (May 8th) and I expect that this can be arranged in May. This would make sure we have one less branch to worry about. We could then just upload trunk, branches/1.0 and tags/1.0 to the collective. That’s as clean as you can get.

The howto “SVN repository Import for Unix users” [Bai] seems like a good plan de campagne for the move from Zest to the collective. Some remarks can be made though.

First of all, this how-to covers the use case of a Zope product that is not yet in any subversion repository. eXtremeManagement of course is. That means we can just do an svn export instead of manual copying. That’s cleaner.

Next to the standard repository layout of trunk, tags and branches, eXtremeManagement also has bundles and integration directories. These are quite large, as several other products are in there (with tar.gz files) and some Zope databases. So we may want to refrain from uploading these directories. That’s something for internal debate though.

The steps noted in this how-to are mostly suited for a product that is not yet in another repository. It will result in just a single revision. If your product is already in a repository, you will want to keep the history, so for instance you know who changed which line in a file. That won’t happen when following this how-to. You really want to make a dump of your current repository and send that to someone at plone.org who can import it for you; probably Wichert Akkerman. I’ll ask on a mailing list whether this is possible.

In answer, Tom von Schwerdtner wrote:
I don’t remember the exact programs, but IIRC you want svn-dump and some filter program so you only get the appropriate “folders” from svn (and nothing that you don’t want to import into the collective). You’ll have to do some googling to figure out the rest.

Once you have the filtered dump file someone with admin access can import it into the collective.

So that is a good sign.
Chapter 5

Conclusion

5.1 What is the status of the tool?

The eXtremeManagement tool is in daily use by Zest Software and other companies (and me personally). It has been released to the general public. The goals of this practice have been met. Other users and developers have filed bugs, added a Czech translation and contributed technical improvements.

5.2 Which improvements can be made to the tool?

There are loads of small improvements that can be made. Let’s concentrate on a few big ones. The eXtremeManagement tool can be moved to the Plone subversion collective. It can be integrated far more with the Poi issue tracker. Progress within an iteration can be made clearer.

5.3 What happens after the practice?

Both Zest Software and I are happy with the way things go, so I continue to work there. I’m taking the “dual route” at the Hogeschool Rotterdam, which means I work four days here and study one day at school, just like the past months. Work on the tool will mostly be done when there is time left. Since we use it extensively we may need to make some time for that every now and then; good tools need sharpening.

My personal conclusion: eXtremeManagement is a good project management tool; the Plone and Zope combination is a good platform to develop on; Zest Software is a fun and inspiring place to work at. Hurray for Free and Open Source Software!
Bibliography


Appendix A

Makefile


GENERATED =

          lot *.aux

# Default target:
default: $(DISTRIBUTE)

report.pdf: preambule.tex start.tex mainpart.tex \ appendices.tex $(GENERATED)

preambule-nl.tex: preambule.tex
    touch preambule-nl.tex

preambule-en.tex: preambule.tex
    touch preambule-en.tex

start.tex: preface.tex summary.tex front.pdf
touch start.tex

mainpart.tex: introduction.tex fixes.tex releasing. \ newfeatures.tex \ conclusion.tex xm.bib
touch mainpart.tex
introduction.tex: images/stories-iteration-10.png
    touch introduction.tex

fixes.tex: images/project.jpg images/iteration.jpg images/story.jpg images/task.jpg images/task-estimate.png images/task-tgv.png images/bookings-per-month.png images/booking-details.png images/task-bookings.png images/portlet-project-administration.png
    touch fixes.tex

ewfeatures.tex: images/stories-iteration-11.png images/progress.png
    touch newfeatures.tex

appendices.tex: Makefile announcement.tex clientwork.tex reportproject.tex colophon.tex
    touch appendices.tex

reportproject.tex: images/xm-report-project.png images/xm-report-content.png images/xm-report-structure.png
    touch reportproject.tex

front.pdf: front.tex preambule.tex images/zestlogo.jpg images/xp-snowboard.jpg

report.pdf: report.tex
    pdflatex $<
    bibtex report
    pdflatex $<
    pdflatex $<


%.pdf: %.tex
    pdflatex $<
    pdflatex $<
%.html : %.tex
   latex2html -no_auto_link -split 3 -no_subdir ←
       -math $<$

##### CLEANING #####

clean:
   rm -rf $(TEMPORARY)

distclean: clean
   rm -rf $(GENERATED)

clobber: distclean
   rm -rf $(DISTRIBUTE)
Announcement

*eXtremeManagement was announced on February 27, 2006 on the plone-users list and the eXtremeManagement mailing list. For the original, see http://article.gmane.org/gmane.comp.web.zope.plone.xm/2.*

Subject: Announcement: eXtremeManagement 1.0

Hi everyone,

Zest Software is proud to announce: eXtremeManagement 1.0, a Plone product for project management using eXtreme Programming techniques.

Zest Software is a software development company based in the Netherlands which is focused at delivering high quality web application software based on Zope and Plone. Zest has been using the eXtreme Programming methodology with success for more than one year by:

- writing tests
- adhering to coding standards (XHTML, CSS2, WAGC, PEP-8)
- programming in pairs
- delivering in iterations of two to three weeks
- involving the customer in the development process.

We developed the eXtremeManagement tool to support our development team and to provide our customers with a unique way to monitor the progress in real time. We have been enthausiastically using this tool in production for almost two months now, and get a lot of positive feedback from our customers.

How does it work?
Managers can add Projects and give a customer permission to view anything in that Project. Then you add an Iteration, a period of usually two or three weeks. Within that Iteration, the customer can add a Story, a feature that they want to have functioning at the end of the iteration. They submit it for estimation and now the employees can give a rough estimate of how long it will take to implement that Story.

At an Iteration planning the customer and the project manager decide which stories really need to go in the next Iteration and which should be postponed, based on what is important to the customer and how much time the development team can put into it. The employees then add small Tasks to each Story and estimate them, so the rough estimate of the Story can be revised. If you assign a Task to someone else, that person will get an email.

When all this is done the Iteration can be started, which will activate the Stories and Tasks, showing the employees that they have work to do. They can book their worked hours in the tool, so the Manager knows how many hours were actually put into the Iteration, so these can be billed to the Customer.

That’s it in a nut shell. More can be learned from the tutorial at plone.org: http://plone.org/products/extreme-management-tool/documentation/tutorial/extreme-management-getting-started

The tool runs on the Plone 2.1 series. It depends on Archetypes 1.3, Poi, DataGridField and AddRemoveWidget.

More info can be found the the plone.org product page: http://plone.org/products/extreme-management-tool

There you will also find pointers to our mailing list (also available at gmane.comp.web.zope.plone.eXtremeManagement), bug tracker, etcetera.

On behalf of Ahmad Hadi, Jean-Paul Ladage and the rest of the team at Zest Software,

Maurits van Rees,

Release Manager

(irc nick: maurits)
Appendix C

Client work

Aside from eXtremeManagement I also did a good measure of work for clients. I'll give a very broad overview of what I made and what I learned.

C.1 Marco Polo platform

Platform that informs and brings together companies and other interested parties that deal with the government. See http://marcopolo-platform.nl.

- Improve the Plone PasswordResetTool.
- Make translations.
- Customise Ploneboard (mostly workflows).
- Customise forms for adding members.

C.2 MinaRaad


- Create an overview of subscribers to content.
- This involved learning about Five (which lets you use Zope3 stuff in Zope2);
- and diving into HTTP requests and sessions.
C.3 Philips Research ICT

Internal (though also international) website of the IT department of Philips at Eindhoven. Not reachable from outside, so no url.

- Find all items owned by a certain user, and change the ownership to another user. This involved “monkey patching” standard Plone functionality, which is cool.

- Make CMFLinkChecker compatible with Plone 2.1 and our products.

- Write link retrievers for PloneHelpCenter.

- Let lms (Link Management System) work via the mandatory Philips proxy. This involved studying the Twisted web server framework in more depth than expected.
Appendix D

Project: practice report

On my local computer I made a Zope instance and installed eXtremeManagement on it. I used that for several personal projects. One of them was of course the project of writing this report on my practice. What better way to end this report than with a few screen shots that illustrate what I have been doing for this report and how long it took.

Figure D.1 on the next page shows the project view of the ‘XM report’ project. Iteration ‘structure’ has been completed already, so iteration ‘content’ is more prominently shown.

Figure D.2 on page 48 then shows the iteration view for ‘structure’. All stories and the iteration have been completed.

Lastly figure D.3 on page 49 shows the iteration view of ‘content’. All stories except two have been completed. The to-do tasks for those stories are listed at the bottom. One of them is of course: adding screen shots for this very section. After that only printing and binding are left.
XM Report
I have to write a report on my practice at Zest Software, where I work on the eXtreme Management tool.

## Iterations

<table>
<thead>
<tr>
<th>Story</th>
<th>Progress</th>
<th>Tasks Open</th>
<th>Tasks Completed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design front page</td>
<td>100 %</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Preface</td>
<td>100 %</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Abstract</td>
<td>100 %</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>100 %</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2: Release 1.0: getting it ready</td>
<td>100 %</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 3: Releasing a Plone product</td>
<td>100 %</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Chapter 4: Release 1.1: new features</td>
<td>100 %</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Conclusion</td>
<td>100 %</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Bibliography</td>
<td>100 %</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Appendices</td>
<td>33.3 %</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Make copies</td>
<td>0.0 %</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>
Figure D.2: eXtremeManagement report: structure iteration
## Content

Add the real content to the report

<table>
<thead>
<tr>
<th>man hours</th>
<th>start date</th>
<th>end date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2006-03-20</td>
<td>2006-05-19</td>
</tr>
</tbody>
</table>

## Stories

<table>
<thead>
<tr>
<th>story</th>
<th>status</th>
<th>estimate</th>
<th>actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design front page</td>
<td>completed</td>
<td>3:00</td>
<td>2:15</td>
</tr>
<tr>
<td>Preface</td>
<td>completed</td>
<td>1:30</td>
<td>1:00</td>
</tr>
<tr>
<td>Abstract</td>
<td>completed</td>
<td>1:00</td>
<td>0:30</td>
</tr>
<tr>
<td>Chapter 1: Introduction</td>
<td>completed</td>
<td>8:00</td>
<td>4:30</td>
</tr>
<tr>
<td>Chapter 2: Release 1.0: getting it ready</td>
<td>completed</td>
<td>7:30</td>
<td>8:30</td>
</tr>
<tr>
<td>Chapter 3: Releasing a Plone product</td>
<td>completed</td>
<td>11:00</td>
<td>11:00</td>
</tr>
<tr>
<td>Chapter 4: Release 1.1: new features</td>
<td>completed</td>
<td>8:30</td>
<td>7:15</td>
</tr>
<tr>
<td>Conclusion</td>
<td>completed</td>
<td>1:00</td>
<td>1:00</td>
</tr>
<tr>
<td>Bibliography</td>
<td>completed</td>
<td>2:00</td>
<td>3:00</td>
</tr>
<tr>
<td>Appendices</td>
<td>in-progress</td>
<td>4:30</td>
<td>1:30</td>
</tr>
<tr>
<td>Make copies</td>
<td>in-progress</td>
<td>3:00</td>
<td>0:00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td></td>
<td><strong>51:00</strong></td>
<td><strong>40:30</strong></td>
</tr>
</tbody>
</table>

## My to-do tasks

<table>
<thead>
<tr>
<th>story</th>
<th>title</th>
<th>estimate</th>
<th>actual</th>
<th>difference</th>
<th>status</th>
<th>assignee(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Make copies</td>
<td>Bind it and give away the copies</td>
<td>1:00</td>
<td>0:00</td>
<td>-1:00</td>
<td>to-do</td>
<td>maurits</td>
</tr>
<tr>
<td>Appendices</td>
<td>Add screenshots of xm report project</td>
<td>0:30</td>
<td>0:00</td>
<td>-0:30</td>
<td>to-do</td>
<td>maurits</td>
</tr>
<tr>
<td>Make copies</td>
<td>Print thrice</td>
<td>2:00</td>
<td>0:00</td>
<td>-2:00</td>
<td>to-do</td>
<td>maurits</td>
</tr>
</tbody>
</table>

Figure D.3: eXtremeManagement report: content iteration
Appendix E

Colophon

This report was written in \LaTeX. The pdf files are made with \texttt{pdflatex}. The editing of texts was done with GNU Emacs. The used browser was mainly Firefox. Screenshots have been made with Gimp. This report was held logically together by make. This all happened on computers with as operating system either Debian ‘‘Sarge’’ or Ubuntu Debian ‘‘Breezy Badger’’.

The file \texttt{preambule.tex} contains the most important settings for \LaTeX:

\begin{verbatim}
\documentclass[12pt]{report}
\usepackage{a4}
\author{Maurits van Rees}
\newcommand{\pag}[1]{(p.~#1)}
\newcommand{\skippart}{(\ldots)}
\usepackage{graphicx}
\usepackage{picins}
%\usepackage{longtable}
%\setlongtables
%\usepackage{pdflscape}
\usepackage{url}
\usepackage{verbatimfiles}
\usepackage{tocloft}
%\usepackage{color}
\author{Maurits van Rees\}
Student number 0545701\}
Class inf5a/dua6a\}
Hogeschool Rotterdam\}
\title{Got Zest?\}
  eXtreme Management of projects in Plone\}
\newcommand{\AGX}{ArchGenXML}
\end{verbatim}
The report calls the file `preambule-en.tex`, which reads the above file and puts on some English specific settings:

\input{preambule}
\usepackage{varioref}