September 1, 2010
ARCHITECTURAL DESIGN DOCUMENT
Software Engineering Project Course
Parempi-group
University of Helsinki
http://cs.helsinki.fi/group/ohtu/

Based on ‘‘HP Architecture Template for Documenting Software and Firmware Architectures’’
http://cs.helsinki.fi/group/kuje/misc/HP_arch_template_vers13_withexamples.pdf

This document is available in pdf-, dvi- and txt-formats.
Also the source of the .tex-file is available at
http://parempi.googlecode.com/svn/trunk/architecturalDesignDocument/

Contents

1 Introduction: Architectural Overview

Parempi-group produced an additional renderer to the group of renderers already available in plasTeX. Our parempi-renderer renders .tex-files into .txt-files with desired format. The produced .txt-file has a format, which is intended to be more readable than a .tex-file for a human user. Especially our aim was to create a .txt-file, which could be comfortably read using a Braille output device.

We have left as much as possible configurable. The user has the possibility to, for example, configure text width, command format, output to several files, handling of unidentified commands and the format of mathematical symbols. A large part of configurations were already available in PlasTeX, and we added a few. The installation and use instructions document also includes a simple scenario, which shows how to write a new function to the parempi-renderer in order to handle an unidentified command.

2 System Purpose: Requirements

The customer provided us with a list of different requirements. On September 1, 2010 one was still able to obtain these requirements in finnish from customer's website [1]. One can also find the same requirements in finnish from below.

2.1 Functional Requirements

The final requirements for each sprint were following:

```
Sykli 1:
    * section/subsection...
          o tällä hetkellä osaa vain ‘‘toisen asteen’’ eli 1.1 subsectionit,
            mutta ei sitä pidemmälle (ei siis osaa 1.1.1)
    * listat (itemize, enumerate)
          o sisennyksen muuttaminen (tehdään nyt välilyönteinä sekä
            tabilla ja laitetaan
          o ini-tiedostoon inttinä; myöhemmin tarkastellaan jos lisätään
            tabulator mukaan myös jollain tavalla)
          o väli bullet pointin jälkeen konfattava
          o miten pistenäyttö esittää bulletpointin?
    * konfigurointitiedosto

Sykli 2:
    * dokumentointi monessa muodossaan
    * defaultiksi konfiguraatio, joka ei hukkaa informaatiota (mm. korostus),
       aggressive tai passive
    * babel? (tktltiki-pohja avuksi muotoiluun/suomennoksiin joskus myöhemmin?)
    * node tutuksi
    * listan ominaisuuden muokkaaminen (bugi)

    * tekstin leveys (rajoittamaton rivinpituus), emph, lihavointi,
    * kappalejako konfiguroitavaksi: sisennys vai ei.
    * taulukot
          o selvitä mitä plastex nyt tekee taulukoille
          o aloitetaan tabular-ympäristöstä
          o hline ja vline -ominaisuudet
          o uusiokäytä array-funktiota

Sykli 3:
   1. sisällysluettelo(toc-tiedosto)
```

[1] http://hlub.dy.fi/ml

```
        1. tiedoston luku
        2. sisällysluettelon kirjoittaminen
   2. kuvat
        1. tiedoston nimi
        2. sisennys
        3. kuvateksti
        4. includegraphics ylikirjoittaminen?
   3. quote(toimii jotenkin)
        1. confattavaksi
        2. sisennys pois
   4. taulukot (joskus ehkä sitten htb?)
        1. hline
        2. vline
        3. rlc
        4. ||
        5. =
        6. cline
   5. konfit (säännöllisiä lausekkeita?)
        1. ei saisi hukata
   6. haxtex (tehty)

Sykli 4:
    * table-ympäristö
    * keskitys pois (konfiguroitavaksi)
    * sisällysluettelon sisennys
    * matikkaa (ascii math notaatio oletuksena)
          o debuggeri
          o esimerkki kurssisivulta:
            http://www.cs.helsinki.fi/group/ohtu/ke-2010/latex.html
    * label/ref (ilmeisesti toimii jotenkin?)
    * informaatiota hukkaamaton versio
          o ignorointi
          o tunnistamattomien komentojen tulostus halutulla notaatiolla
                + notaatio latexmainen: \begin{komento} ... \end{komento}
                jos vaan onnistuu
    * sivut tekstitiedostossa, konfiguroitavaksi (vrt pdf), tavoitteena
      ensisijaisesti kappaletarkkuus
          o latexista lähtevä
                + labelien lisäys lähdekoodiin tarpeeksi tiheästi
                + latexin valmiiden paragraphien käyttö
          (http://en.wikibooks.org/wiki/LaTeX/Document_Structure#Sectioning_Commands)
          o pdf:n lukeminen
          o dvi:n lukeminen (dvitype wikipedia)
          o tietääkö plastex sivunumerot?
          o ulkonäkö:
----sivu 1----
.
.
.

----sivu 2----
    * lähdeviitteet (lähdeviitteiden notaatio, huom. alaviitteet
      oletusarvoisesti samanlaisia plastexin käsittelyn jälkeen)
    * alaviitteet kappaleen (paragraphin) perään
          o mites numerointi? kappaleen sisällä numeroituina vai koko
            dokumentin sisällä?
```

```
          o notaatio {fn nro ...}

Syklit 5:
    * matematiikka
          o array-ympäristö matematiikassa
          o align-ympäristö
          o left ja right aaltosulkuina
          o displaymath
          o math (jos kappaleen ainoa lapsi --> poista dollarit)
                + konfiguroitavaksi jos ei ole kappaleen ainoa lapsi
          o text-ystö matikka-ystössä
          o konfit (ascii math / hlub)
          o matematiikan symbolit
    * pageref (uudet nimet labeleille?)
          o konfiguroitavat nimet?
    * kalvoympäristöt(perus slide generoidut numerot, beamer?)
    * koodin refaktorointi

sykli 6: (LAMA-moniste, muista ajaa mps.tex)
    * dokumentointi (pydoc, käyttöohjeet, ...)
          o esimerkkejä
                + jatkokehitys: tuntemattoman komennon lisäys
                      # tuntemattoman komennon tunnistus
                      # tuntemattoman komennon toteutus
                            * symbolien lisäys (.latextextsymbols)
                            * komentojen lisäys
                      # rekursio: unicode, default
                      # esim. tabular selitys
          o tekstimuoto
          o kartta?
          o debugger
    * refaktorointi --> koodin kommentointi
    * makrot? toimiiko ja miten siellä voi olla?
    * underscore
    * kalvojen numerointi
    * tabular-ystö toimimaan lama-monisteen kan
```

The original requirements from the customer for the first sprints in priority ordering were:

```
1:

- section/subsection...

- listat (itemize, enumerate)

* konfiguraatiotiedosto

2:

- tekstin leveys

- kappalejako: sisennys vai ei

- sisällysluettelo
```

- label/ref

- kuvat

- taulukot

- lähdeviitteet

3:

- matematiikka

- oma conf tiedosto

More specific format requirements were originally:

* yleisiä huomioita

ei tavutusta

riville tulevien merkkien maksimi määrittely mahdollista

sisennystyyli

- joko ei sisennystä ja väli tai

- sisennys ja ei väliä

- ts. noudatetaan kirjoittajan käytäntöä

*dokumentin alku, esim:

```
\documentclass[a4paper,10pt]{article}

\usepackage[finnish]{babel}

\usepackage{graphicx}

\graphicspath{{kuvat/}}

\setlength{\parskip}{1ex}
```

oletusarvoisesti pois

*nimikesivu

```
\author{Harri Laine ja Matti Luukkainen}

\title{Ohjelmistojen mallintaminen}
```

title: Ohjelmistojen mallintaminen

author: Harri Laine ja Matti Luukkainen

*sivurajojen merkintä

käyttäjän määrittelemä merkkijono + sivunumero

oletusarvo page NNN

*sisällysluettelo

1 johdanto 1

2 uml 2

2.1 luokkakaavio 4

2.2 xxx 7

(eli sivunumero heti perään)

*jako tiedostoihin

ei noudateta latex-sorsan tiedostojakoa

\include{osa1}

määritellään itse, esim. tiedosto/luku

* luvut ja aliluvut

\section{Johdantoa ohjelmistotuotantoon}

\subsection{Vaatimusmäärittely}

1. Johdantoa ...

1.1 Vaativuusmäärittely

* listat

\begin{itemize}

\item opetushallinto voi syöttää kurssin tiedot järjestelmään

\item ...

\end{itemize}

- opetushallinto

- teksti alkaa

teksti jatkuu

konfiguroitava listamerkki

* sisäkkäinen lista

```
\begin{itemize}

\item opetushallinto voi syöttää kurssin tiedot järjestelmään

\begin{itemize}

\item sisäkkäistä asiaa

\item ...

\end{itemize}

\item ...

\end{itemize}

- opetushallinto

- sisäkkäistä asiaa

- ...

- lisää tekstiä

* numeroidut listat

\begin{enumerate}

vastaavatsti:

1. aaa

2. bbb

*quote

tekstiä

\begin{quote}

sisennettyä tekstiä

\end{quote}

tekstiä

sisennettyä tekstiä

* sisennys

konfiguroitavissa

onko sisennys aina sama? (itemize, sisällysluettelo)
```

*muotoiltu teksti

{\em tekstiä} \emph{tekstiä} \textem{tekstiä}

muotoon {em tekstia}

{\bf teksti}

* tavutusohjeet

ta\-vu\-te\-taan

poistetaan

* ignoroitavia komentoja

\mbox

\newpage

\pagebreak

\clearpage

\nopagebreak

\setlength

\flushright

\flushleht

\center

\large, \tiny ym tekstin kokoon vaikuttavat

* suorat muunnokset

\ldots   ...

\\   \n (rivinvaihto)

\verbatim

* alaviite

teksti \footnote{}

laitetaan sivun loppuun, ilmaistaan tekstissä kohta johon fn tulee, että alla on footnote

teksti {fn1}

* taulukko eli tabular

\begin{tabular}{ l c r }

```
1 & 2 & 3 \\

4 & 5 & 6 \\

7 & 8 & 9 \\

\end{tabular}
```

```
1 2 3

4 5 6

7 8 9
```

```
\begin{tabular}{ | l | c | r | }

\hline

111 & 2 & 3333 \\

\hline

4 & 555 & 6 \\

7 & 8 & 9 \\

\hline

\end{tabular}
```

```
111 | 2 | 3333

----------------

4 | 555 | 6

7 | 8 | 9
```

```
tai kaikki viivat

multicolumn huomioitava luonnollisella tavalla

\arraystretch

\tabcolsep

ym voidaan ignoroida

* kuvat

ilmaistaan, että kuva {figure 10 kuva.eps}

ilmaistaan tiedoston nimi

kuvateksti säilytetään
```

```
* kuva, sivu ja kappaleviitteet

\label, \ref, \pageref

ekspandoidaan

*asciimath symboolit:
```

Adjusted and more specified requirements for mathemathics were:

```
syntaksi: http://www1.chapman.edu/~jipsen/mathml/asciimathsyntax.html

* sulut

käytetäänkö aina samoja sulkuja operaattoreiden parametreissa vai

vaihteleeko tilanteen mukaan

* välien poisto

(a+1) (a+2)  -> (a+1)(a+2)

( a + 1) -> (a + 1)

( a + 1 ) -> (a+1)

a +     2 = 3 -> a + 2 = 3

eli monta väliä -> 1 väli

*symbol

latex            hlub            asciimath

-----------------------------------------

\forall            \A \all

\exists            \E \exists

\lor \vee       || or

\land \wedge    && and

\in                \in (in)

\geq            >=

\emptyset        \0

2^{n+1}            {} tai ()

\cup            \U \union
```

```
\cap            \I \intersets

\setminus         \M \setminus

\subset            \sub

\subseteq        \sube

\supset            \sup

\times             \X

\equiv             ==

\neq            !=

muunneltu fontti kirjaimissa:

\mathbb{R}        \R

\mathbb{NN}    \NN

muut vastaavasti

kreikkalaiset aakkoset:

\labda             ~l

\alpha             ~a

mathbf käsitellään kuten normaalissa tekstissä

*nuolet

luonnollisella tavalla

\leftarrow      <-

\longleftarrow    <--

\Leftarrow        <==  eli aina tuplana jotta ei sotkeudu vertailuihin

\Longleftrightarrow <=>

harpoonit ja hookarrowit kuten nuolet

*hatut

\hat{XY}            ^XY

\widehat{XY}        ^XY

\bar{a}                ei vielä päätetty

\vec{a}                <a>
```

```
\vec{AB}            <AB>

\overrightarrow{AB}    <AB>

entä muut hattusymboolit?

*parametrilliset operaattorit

\lim_{n \to 10}        lim{n -> 10}

\sum_{i=1}^{n}        sum{i=1 -> n}

vastaavasti joukko-operaatioille

\prod => prod

\frac{n-1}{20}        (n-1)/20 (sulut tarvittaessa)

\dfrac                 samoin kuin edellinen

p_{ij}                p_ij

p_{i,j}                p_{i,j} tai voi muuttaa ylläolevaksi

p^3_{ij}            sama

\sqrt{x+1}            sqrt{x+1}

\sqrt[3]{4}            3rt(4)

a \cdot b            a * b

1 \ldots n            1 ... n

 myös \vdots ja \ddots  merkitään ...

a \bmod b             a mod b

x \equiv a \pmod{b}    x == a (mod b)

\int_0^{10}            int(0,10)

*monimutkaiset operaattorit

\binom{n}{k} =\binom{n-1}{k} + \binom{n-1}{k-1}

bin(n  k) = \b((n-1)  k) + \b((n-1)  (k-1))

a \stackrel{def}{=} 10+x

a ={def} 10+x

\sum^n_{\substack{0<i<n \\

j\subseteq i}} P(i,j) = Q(i,j)
```

```
\sum{0<i<n; j \sube i -> n}

operaattoria tulee edeltää joko välil, rivinvaihto tai sulku

* sulut

\Big((x+1)(x-1)\Big)^{2}

((x+1)(x-1))^2

kaikki sulut normaalikokoisia


*ympäristöt

$e = mc^2$

poistataanko dollarit?

- jos omalla rivillään niin kyllä

- entä jos tekstin seassa: Puun $T$ solmuissa ...

konfiguroitavissa

\begin{equation}

e = mc^2

\end{equation}

kaava (1.1):

    e = mc^2

    jatkuu

ei keskitystä

\begin{equation*}

e = mc^2

\end{equation*}

kaava:

    e = mc^2

\[ e=mc^2 \] toimii kuten equation*

* suorat muunnokset

$e=mc^2 \text{Einstein} \qquad f=ma \text{Newton}$
```

```
erilaisia välejä

\quad  pieni väli -> space

\qquad pitkä väli -> 2*space

\! -> space, myös jos usea \!

* poistetaan

\!

\text{}

* jätetään tekstiin

\underline{teksti}

\overline

* allign ja eqnarray

käsitellään kuten normaalit taulukot, esim:

\begin{align}

a & = b + c

& = d + e + j + k + l \nonumber \\

& + m + n + o \\

& = p + q + r + s

\end{align}

a = b + c                (3.10)

= d + e + j + k + l

+ m + n + o           (3.11)

= p + q + r + s       (3.12)

jos määritelty \lefteqn, tehdään muotoilu ilmeisellä tavalla, tyyliin,

esim:

a+b+c+d+e

 = f + g

 = q + w

\begin{equation*}
```

```
|x|=

\begin{cases} x & \text{if $x\geq0$,}

\\

-x &\text{if $x\le 0$.}

\end{cases}

\end{equation*}

|x| = {x   if x>= 0

     -x  if x < 0}

jos useampia vaihtoehtoja, niin muoto:

|x| = {

    x   if x>= 0

    -x  if x < 0

}

\IEEEeqnarray käsitellään vastaavasti, ks \tabular

\: voidaan jättää huomioimatta

\IEEEeqnarraymulticol{} käsitellään ilmeisellä tavalla

* taulukot ja matriisit

sama asia tehdään joskus seuraavasti:

\begin{equation*}

T(k) = \left\{

\begin{array}{ll}

 O(1) & \text{kun } k=1\\

 T(k/2)+O(1) & \text{kun } k>1

\end{array} \right.

\end{equation*}

joka renderöitävä myös

T(k) = { O(1) kun k=1

       T(k/2)+O(1)  }
```

```
eli taulukot käsitellään luonnollisella tavalla kuten \tabular

\matrix ja sen variantit käsitellään luonnollisella tavalla:

\begin{bmatrix}
1 & 2 \\
3 & 4
\end{bmatrix}

( 1 2 )
( 3 4 )

*teoreema, todistus

\newtheorem{teor}{Teoreema}

\begin{teor}
AVL-puun korkeus on O(log n)
\end{teor}

jos mahtuu yhdelle riville:
Teoreema 1. AVL-puun korkeus on O(log n)

jos ei niin:
Teoreema 1.
  AVL-puun korkeus on O(log n)
  ja syvyys myös O(log n)

HUOM: teoreemien numerointi!

\begin{proof}
oletetaan että ...
\end{proof}

Proof:
  oletetaan että ...   MOT

* mietinnän alla

\underbarce{a+b+c}_{summa}
{a+b+c}_{summa}

\overbrace vast

\multiline ehkä huomioimatta
```

Then the requirements for a certain example material (lama-moniste) were:

```
Konfigurointia?

[environments] // suuren ympäristöt \begin{...}... \end...
```

```
array -> tabular

mycode -> clrscode2e

[singles] // yksinkertaiset komennot

Blue -> remove

Green -> remove

[?????] // muut siltä väliltä

emphasis -> {em \1}

em -> emphasis

bem -> emphasis

rem -> emphasis

seuraavasta ei kannata välittää:

emphasis: {\ []}+} -> {\ \1} (regexp)

em: {\ []}+} -> emphasis

bem: {\[a-z]em []}+} -> emphasis, eli myös esim rem samoin

värit: \Blue, \Green -> remove


kivisen makrot: // tiedostokohtainen konf

\bem -> \em

\Blue -> poistetaan

\begin{mydesc} -> normaalisti

\begin[slide} -> s1,s2,s3,...

\set{ ... } -> { ... }

\mid -> |

S \derives v_1 muotoon S -> v_1

S \dderives v_1 muotoon  S => v_1

a \leadsto b -> a leads b

\begin{mycode} -> "hyvin", eli kuten \usepackage{clrscode3e} tai 2e

\vname{noun-phrase} ->  <noun-phrase>
```

```
\tname{hamburger} -> hamburger

\newcommand{\derTMP}[2]{$\proc{Derives}(#1,\mathrm{#2})$}

\derTMP muotoon Derives

muut:

q_0\stackrel{{\Blue 0}}{\rightarrow}q_1

muodossa: q_0 ->{0} q_1

\esmove ja \esmovec kuten leadsto+stackrel

$\overline{A}$

muodossa: A^c

\mathrm{} kuten mathbb

\ast -> ^*

\delta ~d

\Sigma ~S

\varepsilon ~e

\Gamma ~G

\bigcup{a \in A} -> kuten \sum

\rma,\rmb,\rmc,\rmd -> a, b, c, d

\underbrace{R * \ldots * R}_{\mbox{$k$ kertaa}}

muotoon: {R * ... * R}_{k kertaa}

\rm -> pois

\vdash merkataan  |-

\blank merkataan ! (konfiguroitava, dokumenttikohtainen)

\hash merkitään #

\code{xx}  ->  <xx>
```

2.2 Non-functional Requirements

Parempi-renderer is expected to work in an execution time that is at most a few minutes always depending on the length of the input, i.e. the .tex-file. The output to a .txt-file should remain the same for all executions of the program using exactly the same .tex-file. The program should also be such, that a normal university computer science student is able to install and run the program using the instructions

provided. The format of a rendered .txt-file should be such, that it can be read using a Braille output device. We were also insisted to document the whole program so, that becoming acquainted with the program should be easier than it was for us with plasTeX.

3 Structure

In the following we summarize the structure of the program created.

3.1 Overview: overall structure

Below is a process diagram of all components and their collaboration with each other. It shows the process of rendering a .tex-file into a .txt-file using the components needed. The diagram below the first one shows how LaTeX, PlasTeX and Parempi-renderer are interlinked. After that one is presented with a list of different components we implemented in PlasTeX.

```
1. file.tex -> LaTeX => file.aux + file.toc + etc
2. ref.beb -> BibTeX => file.bbl
3. file.tex + file.aux + file.toc  + file.bbl -> PlasTeX + Parempi-renderer => file.txt
```

```
LaTeX - Packages - PlasTeX
           |          |
         Parempi-renderer
```

3.2 Components

- parempi renderer: __init__.py
  - located in plasTeX/Renderers/Parempi/
  - renders .tex-file into a .txt-file
  - renderer goes through the document tree formed by plasTeX's parser and renders the tree into a .txt-file
  - uses .plasTeXrc, .latextextsymbols
  - subclass of Renderer in plasTeX.Renderers
- .plasTeXrc
  - config-file with ini-format (see `http://en.wikipedia.org/wiki/INI_file`)
  - located in user's home directory
  - read in the beginning of rendering process in init.py using plasTeX.ConfigManager
- .ParempiMathrc
  - math config-file with ini-format
  - located in user's home directory
  - read in the init.py using plasTeX.ConfigManager if math-environment is used in file.tex
- slides.py
  - located in plasTeX/Packages/
  - used in __init__.py if in .tex-file 'documentclassslides'
  - defines command 'beginslide' ... 'endslide' to be an environment, this enables parser to parse this command properly

- parempi debugger: __init__.py
  - located in plasTeX/Renderers/debugger/
  - ouputs to a file named after 'fileX.tex' as 'fileX'
  - output shows the document tree formed by plasTeX's parser

4 Dynamic Behavior

Basically the only dynamic behavior is to render some .tex-file into .txt-file. However, there are some modifications one can do to change both the ouput to and format of the rendered .txt-file. All of these different usecases are listed and abbreviated below. One can find the same examples under usecase examples in the manual.
Creating a txt-file from a tex-file:

- in this scenario one uses only PlasTeX and its classes
- simply type 'plastex file.tex'
- in this case it is important that your config-file is the default found from repository, otherwise you might not use Parempi-renderer as a default
- if Parempi-renderer is not your default-renderer, then you can use by typing 'plastex –renderer=Parempi file.tex'
- after running Parempi-renderer you should find the output in the same directory named as 'file.txt' if you are using default configuration provided

Using debugger

- type 'plastex –renderer=debugger file.tex'
- open output for example by typing 'emacs file' (note that the debugger outputs a file without any type-mark)

Adding table of contents

- in this scenario one uses both LaTeX and plasTeX
- Note, that in the beginning of the .tex-file one has to include a row, which says '\tableofcontents'. Otherwise no file.toc will be generated using LaTeX and no table of contents will be generated by using PlasTeX. PlasTeX uses LaTeX generated .toc-files to generate a similar table of contents as LaTeX created.
- first type 'latex file.tex'
- second type 'plastex file.tex'

Adding references

- in this scenario one uses both LaTeX and plasTeX
- note, that in the end of the .tex-file one has to include a row, which says '\bibliography{references}' or otherwise no file.aux with references will be generated using LaTeX and no references using PlasTeX
- in addition one has to have references.bib in this case in the same directory with the file.tex with proper LaTeX bib-file format
- first type 'latex file.tex'
- second type 'bibtex file'

- third type 'plastex file.tex'

Changing config file

- in this case just go to your home directory and find .plasTeXrc or .ParempiMathrc and edit the file
- The file has an INI-format although it is not named as .ini-file. You can see more about the INI-format for example fro m wikipedia (see `http://en.wikipedia.org/wiki/INI_file`)
- below is a one example of how to do changes to the INI-files

Changing emphasize format in .plasTeXrc

- first look for .plasTeXrc in your home directory and open the file using some text editor
- next change the text on the line starting with emphasizeFormat after =-mark for example as '\em{ %s}'
- what you get out after this change is \em{text} for \emph{text}
- remember to save your changes to the file before next time using PlasTeX in order to see the changes

Overriding a command in file.ini

- one can define commands to be overrided in file.ini

5 Other Views

5.1 Design views and their justifications

We decided to implement a renderer into PlasTeX instead of creating an entirely new program. This allowed us to concentrate in building the renderer and especially in how to render .tex-file so, that it is as readable as possible using a Braille output device. This means that anybody using our renderer should also get acquainted with PlasTeX as the rendering process is entirely based on the PlasTeX document structure. Using PlasTeX enabled us also to use the wide variablity of different command line parameters already implemented in PlasTeX.

Choosing PlasTeX as our basic framework helped us also further in smaller details. PlasTeX for example provided us with the ConfigManager, which offered us a way to easily read files with INI-format. This meant that we decided to make all our configuration files INI-files. Then we were able to concentrate on the configuration specifications, which were mainly derived from the needs of the customer (like overriding commands, using

PlasTeX also had a language file i18n.xml, which defines language specifications for a few different languages. We decided to use this same structure. It uses package Babel, which is originally a LaTeX package, but implemented in PlasTeX as a python package Babel in directory 'plasTeX/Packages/babel.py'. This enables our program to be capable to use any language, if the language is just defined in i18n.xml using UTF8-encoding and xml-format. Using UTF8 in i18n.xml was also mainly derived from PlasTeX as we had no reason to step away from a very universal encoding.

We also decided to use LaTeX as part of our rendering process. LaTeX generates .toc-files and .aux-files, which are used by our Parempi-renderer. This can be justified as our main purpose was to help our visually impaired customer to be able to read and talk about .tex-files, which are normally rendered into a .dvi- or a .pdf-file. By using LaTeX we were able to obtain information about the .pdf-file generated using the same .tex-file. This means that for example the section, slide and picture numbers should be the same in both LaTeX generated .pdfs and PlasTeX with Parempi-renderer generated .txt-files. This means that for example following lectures might become easier for the visually impaired student as he or she will have the same information about the slides and pictures that appear in lectures.

The customer also asked if we were able to render exactly the same page breaks into the output .txt-file. As the time at our disposal was limited, this task was made easier and we ended up just rendering all different labels as well as possible. The idea of rendering labels was, that for example the command

\pageref}label} could then be rendered so, that the label would also be part of the output. This way the reader would be able to find the text referenced using the labels provided. However, as everybody should understand this is only a compromise and the proper way to do this would be to render the page breaks. After searching for an answer to this question, we ended up with a proposal: One way to do the page breaks almost in the similar way as in pdfs could be to use dvitype unix-program. It outputs the insides of a .dvi-file in a human-readable form. Thus, by mathching words and sentences one could be able to find a way to find almost always correct page numbers to a .txt-file when compared to .dvi-file. Then assuming that the .dvi-file can be changed into a .pdf-file, one could end up with a .txt-file and a .pdf-file with similar page breaks.

Mathematical symbols used are in principal based on ASCIIMath notation. However, as our customer already had many views about good and not so good notations, we tried to leave as much as possible as configurable. Different configurations are meant to help with this issue as well, and finally the developer manual is included with a detailed explanation of how to include new commands and environments into Parempi-renderer when needed.

The footnote notation was developed in cooperation with the customer. It seemed for the custommer that reading the footnotes right after the paragraph where they appeared, was the most comfortable way to keep track of them when page breaks were missing. Most of the other commands we implemented were mainly based on the documentation available for LaTeX. For example we implemented a few different notations for references, and all of them were based on LaTeX notations.

5.2 Process

In the meant use for parempi-renderer there should be only a single instance of it running at once. We have not tested multiple instances at the same time, and suppose strongly that something will break down if multiple instances are used at once.

5.3 Development

Whole codebase is located at `http://code.google.com/p/parempi/trunk`.

5.4 Physical

To run Parempi-renderer one needs installed PlasTeX. Although LaTeX might not be needed, we recommed also its installation as it provides many packages for PlasTeX. Also Python version < 3 should be installed.

5.5 Deployment

All the components are deployed according to the structure of PlasTeX.

6 Test plan and test solutions

Testing was mainly build around compiling latex documents in Plastex, because Plastex doesn't offer sufficient automatic testing tools for Plastex DOM and building testing framework around Plastex DOM could have taken a long time. In the testing process we mainly used documents given by customer and more specifically created one-task-latex documents etc Array.tex.

6.1 Test Design

Testing will be mostly build around specifications given by customer. All compiled latex documents should meet the specifications. Customer gave LAMA-handout,OHMA-handout and TIRA-handout which we examined more closely. Plastex was installed using instructions given by manual, we used plastex version 0.92 and python version 2.6.

6.2 Test Cases

6.2.1 LAMA-handouts

Compile LAMA-handout with plastex and try to find results where compiled text does not comply with specifications.

6.2.2 OHMA-handout

Compile OHMA-handout with plastex and try to find results where compiled text does not comply with specifications.

6.2.3 TIRA-handouts

Compile LAMA-handouts with plastex and try to find results where compiled text does not comply with specifications.

6.2.4 specifically created one-task-latex documents

Created test_tex folder which contains latex documents for explicit tasks. In this case mostly tasks which customer had given before starting a scrum-sprint. Use test script testscript.py which compiles all documents in the test_tex folder, do visual review of how accurately visual representation responds to specifications.

6.3 Test procedure

- make a new latex document or find an old latex document which fits best for the purpose of the test.
- run latex document through renderer.
- make visual review through rendered text.

6.3.1 test Item Transmittal report

`http://code.google.com/p/parempi/source/browse/#svn/trunk/test_tex`

6.4 Incident Report

OHMA-handout did not compile. Problem with the Plastex framework, more specifically with including multiple files and minor problem with verbatim.

Tira-Luukkainen did not compile. Problem with recursion depth. Also a problem with include and a wierd problem with fileNames. see Test Summary Report.

6.5 Test Summary Report

6.5.1 LAMA-handout

Lama-Kivinen did compile. Some minor problems with math symbols and macros.

6.5.2 OHMA-handout

Did not compile. Problem with Plastex Framework, more specifically with including multiple files with bibtex. And problems with verbatim.

```
errors: File "/usr/local/lib/python2.6/dist-packages/plasTeX/Base/LaTeX/Verbatim.py", line 21, in in
escape = self.ownerDocument.context.categories[0][0] IndexError: string index out of range

File "/usr/local/lib/python2.6/dist-packages/plasTeX/Base/LaTeX/Bibliography.py", line 28, in loadBi
file = tex.kpsewhich(tex.jobname+'.bbl') in osa2.tex
```

6.5.3 TIRA-handout

Tira-kivinen did compile.but there were problems in slide 227. TIRA-luukkainen did not compile. There were problems with recursion depth and some problem with including macros. Tira-luukkainen did compile after we disabled macs.text and compiled tira2010.tex in two parts.

```
"/usr/local/lib/python2.6/dist-packages/plasTeX/Filenames.py", line 249, in _newFilename raise Value
```

6.5.4 specifically created one-task-latex documents

All smaller documents did compile. There were some minor problems with math, mostly with indents and symbols.