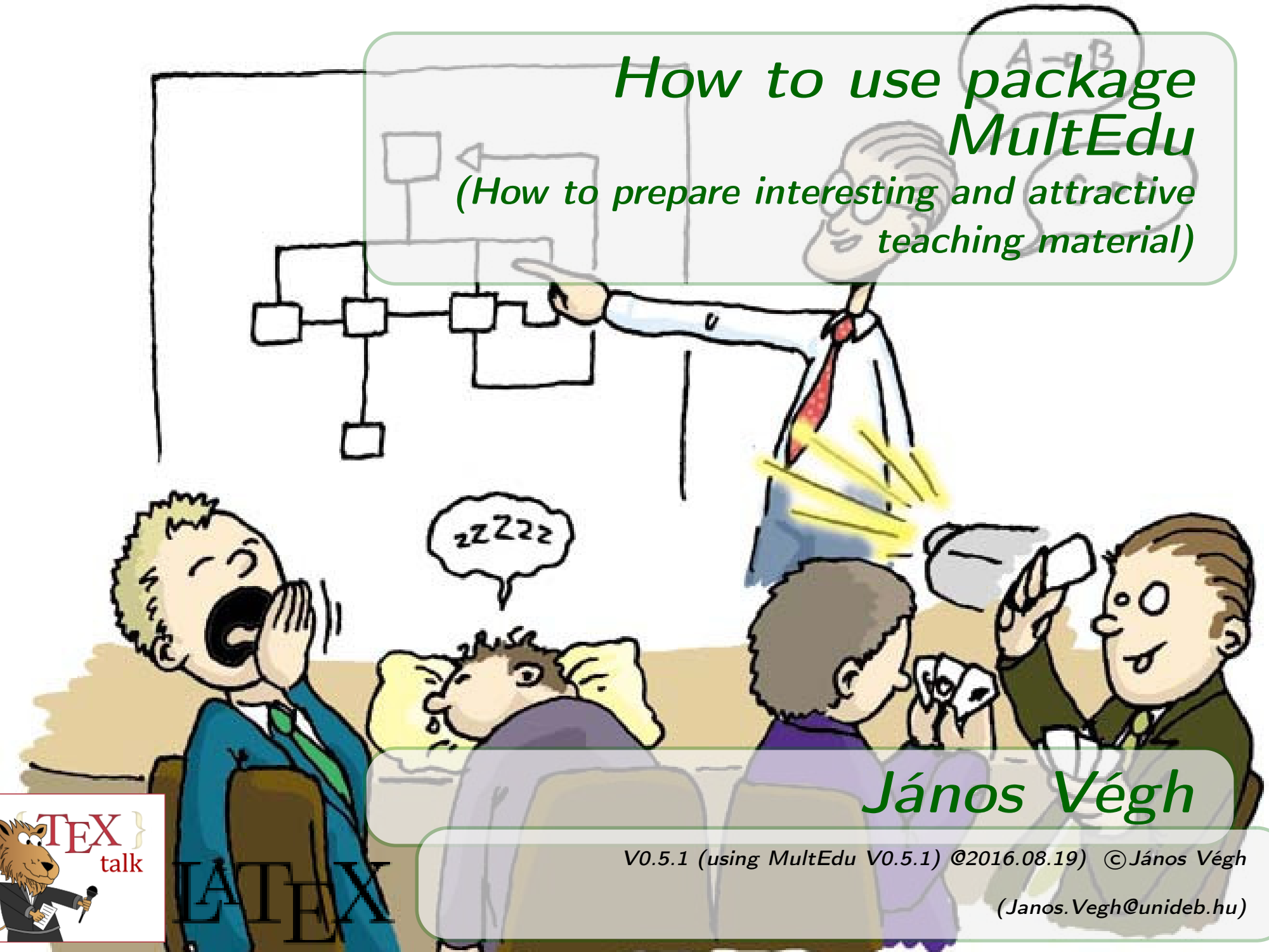
How to use package MultEdu

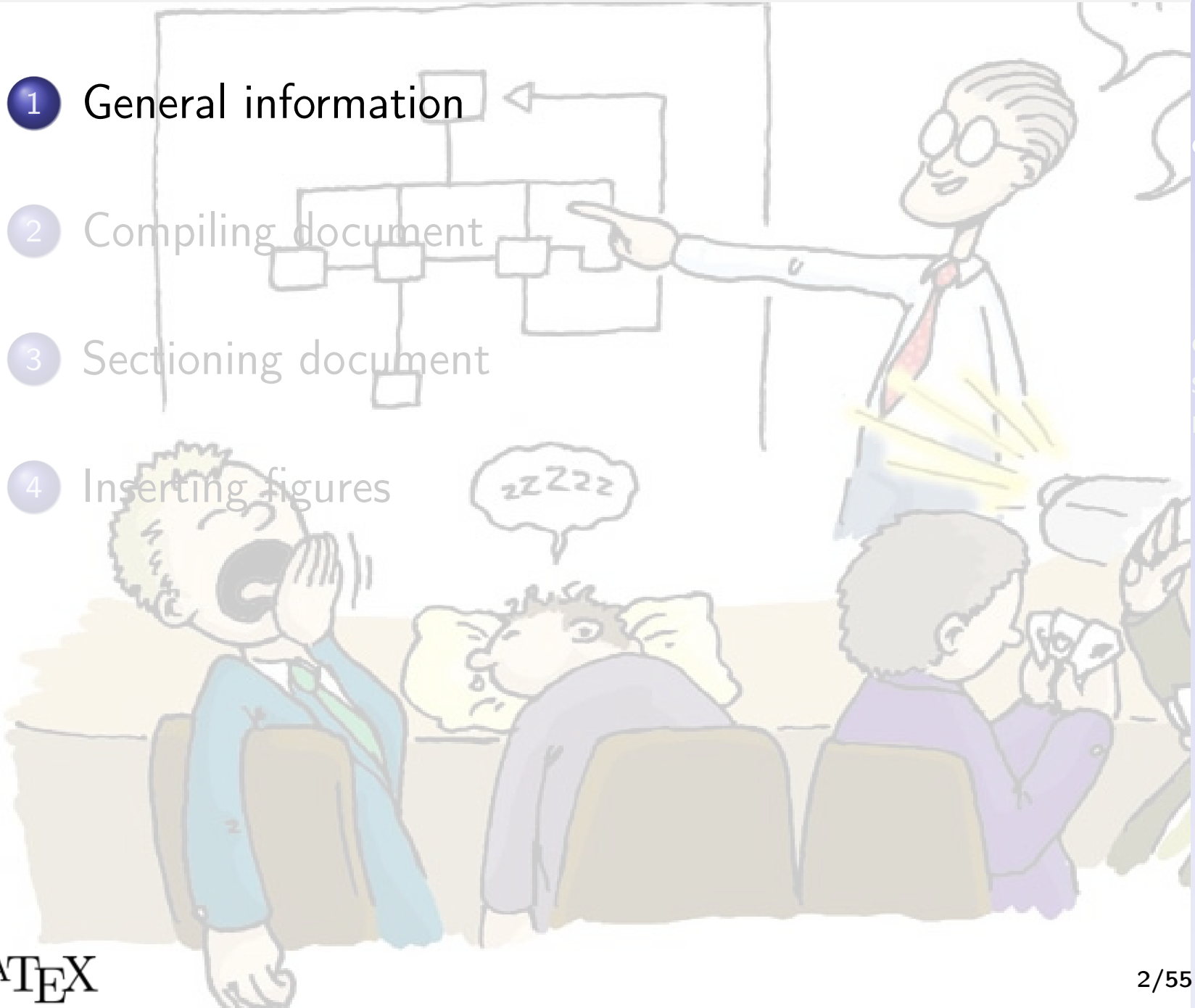
(How to prepare interesting and attractive teaching material)



János Végh

V0.5.1 (using MultEdu V0.5.1) @2016.08.19)

(Janos.Vegh@unideb.hu)



The todays education needs the course material in various forms: in the lecture room for the projected picture well organized text with many pictures are needed, which also serve as a good guide for the lecturer, too. To prepare for the exams, the explanation provided by the lecturer when projecting the slides is also needed. The present document is a demo and test at the same time. It attempts to describe the many features, and also tests if the features really work. Because of the many features, and their interference, this job needs a lot of work and time, so the documentation does not always match the actual features, especially in this initial phase.

The macro package can be used at (at least) three different levels. Even the lowest level assumes some familiarity with LaTeX. At the very basic level, you might just take the package, replace and modify files in the distribution. At the advanced level (this assumes reading the User's manual ☺) the user learns the facilities provided in the package, and prepares his/her courses actively using those facilities. Power users might add their own macros (preferably uploaded to the distribution), i.e. take part in the development.

Multedu, as any package based on LaTeX, assumes that the user has experiences with using LaTeX. I.e. some LaTeX distribution must already be installed on the system of the user. If you want to use the batch processing facility, the CMake system must also be installed.
For the simplicity of utilization and starting up, the best way is to create a main directory for your family of projects and a subdirectory for your first project, as described below. The quickest way is to copy `./Workstuff` (after deleting and renaming some files) and to prepare your own "Hello World" program. Making minor changes to that source you may experience some features of the package. Then, it is worth at least to skim the user's manual, to see what features you need. After that, you may start your own development. At the beginning text only, later you can learn the advanced possibilities.

The MultEdu system is assumed to be used with the directory structure below. It comes with two main subdirectories: `./common` comprises all files of the MultEdu system, and `./Workstuff` models the users subdirectory structure.

```
.
|-- common
|-- WorkStuff
```

You may add your project groups stuff like

```
.
|-- Exams
|-- Labs
|-- Lectures
|-- Papers
```

which directories have a subdirectory structure similar to that of `|-- WorkStuff`

Subdirectory ./common comprises some special subsubdirectories and general purpose macro files.

```
.
|-- common
|  |-- defaults
|  |-- formats
|  |-- images
```

Subsubdirectory ./defaults contains some default text, like copyright.

Subsubdirectory ./formats contains the possible format specification macros, here you can add your own format macros.

Subsubdirectory ./images contains some images, partly the ones which are used as defaults.

Subdirectory ./Workstuff contains the files of the present demo, and serves as an example of using the system (a kind of User's Guide). It contains a sample project ./Workstuff/Demo, which has three main files.

```
|-- WorkStuff
|  |-- Demo
|  .  |-- CMakeLists.txt
|  .  |-- Demo.tex
|  .  |-- Main.tex
```

The real main source file is Main.tex, and Demo.tex is a lightweight envelope to it. (if you want to use UseLATEX, you need to use the file with name Main.tex, the envelop must be concerted with the CMakeLists.txt file)

```
|-- WorkStuff
|  |-- Demo
|  .  |-- build
|  .  .  .  |-- build
|  .  |-- dat
|  .  |-- fig
|  .  |-- lst
|  .  |-- src
```

The file `Main.tex` inputs files in the sub-subdirectories.

Subsubdirectory

| . |-- `src` is the place for the user's source files,
| . |-- `fig` for the images.
| . |-- `lst` for the program source files,
| . |-- `dat` for the other data .

It is also possible to use CMake package UseLATEX for compiling your text to different formats and languages in batch mode; producing the documents in different languages and formats in one single step. File `CMakeLists.txt` serves for that goal.

Subsubdirectories

`|-- build` and

`| . . |-- build`

are only needed if using CMake.

Some kind of heading usually belongs to the document. As an example see file `src/Heading.tex` of this user's guide.

Line \def\LectureAuthor{J\'anos V\'egh} defines the author, lines \def\LectureTitle{How to use package MultEdu} and \def\LectureSubtitle{(How to prepare interesting and attractive teaching material)} the main title and its subtitle. Also a university name or conference name can be defined in \def\LecturePublisher{University or conference} line. It is good practice to define \def\LectureRevision{V\Version\ (using \MERevision) \at 2016.08.19}, too.

When using dual-language source files, one has to prepare the source in a form which allows to select source lines depending on the language. To prepare dual-language documents, the definitions should be put in frame like

```
\ifthenelse{\equal{\LectureLanguage}{english}}
{% in English
}% true
{% NOT english
}
```


Also here you can give e-mail address

```
\def\LectureEmail{Janos.Vegh\at unideb.hu}
```

Furthermore, one can provide BibTeX, even conditionally, depending on the language or the presence of some files

```
\IfFileExists{src/Bibliographyhu}
{\def\LectureBibliography{src/Bibliography
,src/Bibliographyhu}}
{\def\LectureBibliography{src/Bibliography}}
```

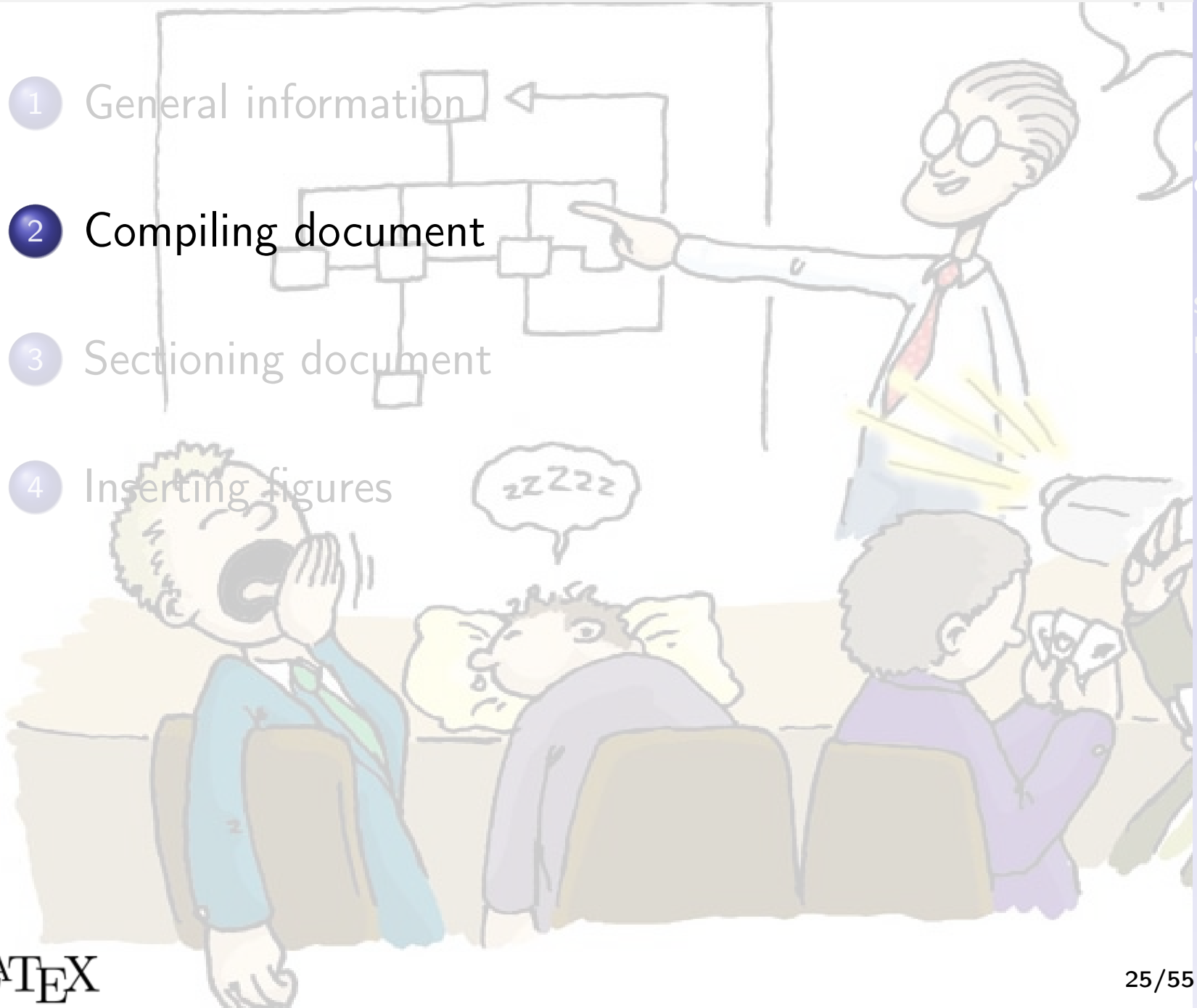

a

Multedu allows to utilize two popular screen width. The default is the spreading format with aspect ratio 16:9. To set ratio 4:3, use

```
{\def\DisableWideScreen{YES}}
```


Sometimes (mainly in the case of short presentations) the table of contents is not necessary at all. It can be disabled through defining
{\def\DisableTOC{YES}}
It might also happen, that chapter-level TOC is still needed, but the section level not. This can be reached through defining
{\def\DisableSectionTOC{YES}}

a



The MultEdu system works perfectly with its default settings, but it cannot read your mind. The settings can be changed using definitions of form `\def{\xxx}`. The place where the settings can be changed, depends on the compilation mode. The next two sections shows the utilization of the compilation modes, while the third one describes the settings in details.

File `Main.tex` is the common part of the dual compilation system. This contains the real source code. Any setting in this file (as well as in the included files) overwrites the settings, in both the manual and the batch mode, so it is better not to use any settings here. The best policy is to collect all the settings in a separate file, which is then included in the envelope file.

Developing course materials is best to do using an editor, integrated into an IDE. You need to read the envelope file (corresponding to `Demo.tex`) into the editor and mark it as your main document. In the file `Main.tex` you should insert references to the chapters of your course material. Those chapter files should be placed in subdirectory `src`, following the structure of the demonstrational material.

The settings file should be placed in subdirectory `src`, its reasonable name can be `Defines.tex`. The task of the wrapper file `Demo.tex` is only to input the setting file and the main file.
The batch compilation generates a file `Defines.tex`, which goes into subdirectory `build/build/src`. (You may use it to 'cheat', what settings and how should be utilized.) The batch compilation also generates a template file `Defines.tex.in` in subdirectory `src`. The content of this file corresponds to the last pass of the batch compilation.

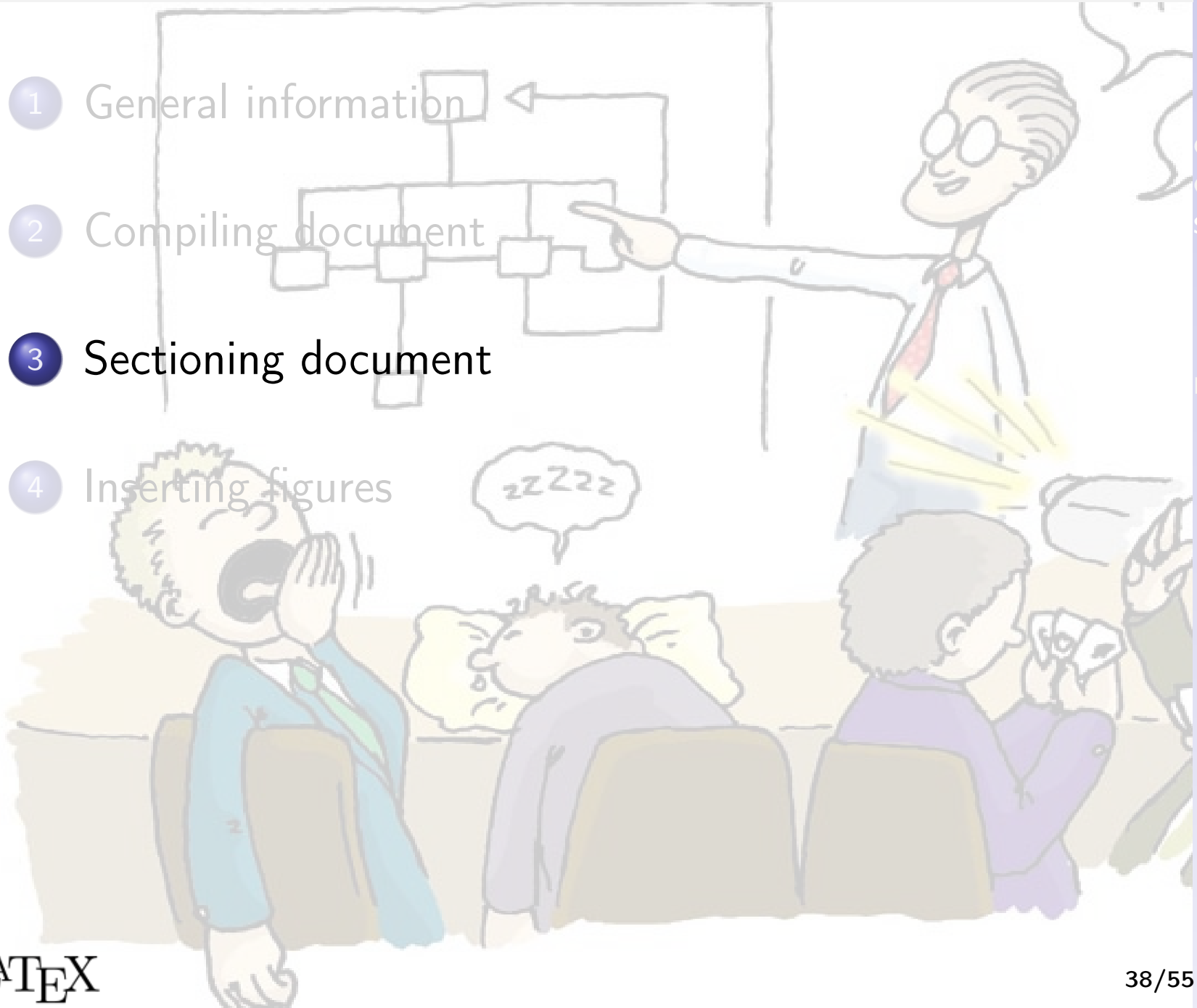
Batch processing serves (mainly) the goal to generate the output from the common source in the different formats and languages.
From technical reasons, MultEdu prepares a private copy from the MultEdu files, in the subdirectory `common` of the project. You may safely experiment with this copy or also delete it; the next batch compile will recreate it.
(I.e. one should save the valuable developments; possibly in subdirectory `../../common` if you want to use it also by the other project groups.)

Multedu uses three-level version numbering (major, minor and patch). The course materials prepared with MultEdu have two kinds of version numbers: the user maintains his/her own version numbers, and the developer maintains version of MultEdu.
Version number of MultEdu is located in file `../../common/MEMacros.tex`; better not to change it.
The own course material version number is held in file `CMakeFiles.txt`, and that setting will be refreshed in the generated source files (through file `Defines.tex`) when batch compiling. The version number of the course material appears also in the name of the generated file, so it is worth to use it in a consequent way.
Usage:

```
\def\Version{major.minor.patch}
```

MultEdu can handle single- and dual-language documents. Different spelling, section name, captions belong to the different languages. In the settings file the language must be specified, like using setting `\LectureLanguage{english}` (this is the default). The name of the selected language appears also in the name of the result file.

In the dual-language documents, a first and second language co-exist, meaning in which order the texts in the different languages appear in the document. This allows to develop course material in both languages simultanously, one below the other. Selecting the proper language one can generate output in either language. If `\UseSecondLanguage{}` is defined, then the text appearing in the second position will be processed, using the language features defined by `\LectureLanguage{}`. When using batch compilation, the options `FirstLanguage` and `SecondLanguage` must be provided (that defines the language found in the dual-language macros in the first and second position, respectively). If option `NEED_BOTH_LANGUAGES` is on, the output file will be produced in both languages. If it is switched off, option `USE_SECOND_LANGUAGE` decides which language to use.



Basically, the document must be organized as 'beamer' needs it, but to print it in a book-like form, the sectioning must be changed, and also the package 'beamerarticle' must be used. In order to provide a uniform wrapper around sectioning, MultEdu introduces its own sectioning units.

These units actually correspond to the ones used in format 'book', and MultEdu transforms them properly when preparing slides.

Usage:

`\MEframe[keys]{subtitle}{content}`

Legal keys are

`shrink=true|false` and `plain=true|false`

By default, both are false.

Correspondingly, the biggest unit is the 'chapter'. (As mentioned, for slides it is transformed to 'section'.)
Usage:
`\MEchapter[short title]{long title}`

The next, smaller unit is the 'section'. (As mentioned, for slides it is transformed to 'subsection'.) Usage:
`\MEsection[short title]{long title}`
In a similar way, there exists
`\MEsubsection[short title]{long title}` and `\MEsubsubsection[short title]{long title}`; the latter one is transformed for slides to `\paragraph`.

It happens, that I teach the same course in my mother tongue for my domestic students, and in English, for foreign students. The course material is the same, and it must be developed in parallel. Obviously it is advantageous, if they are located in the same source file, side by side; so they can be developed in the same action. The \UseSecondLanguage macro supports this method.

The macros introduced above have a version with prefix 'MED' rather than 'ME' only, which takes double argument sets (arguments for both languages). Depending on whether \UseSecondLanguage is defined, the first or the second argument set is used.

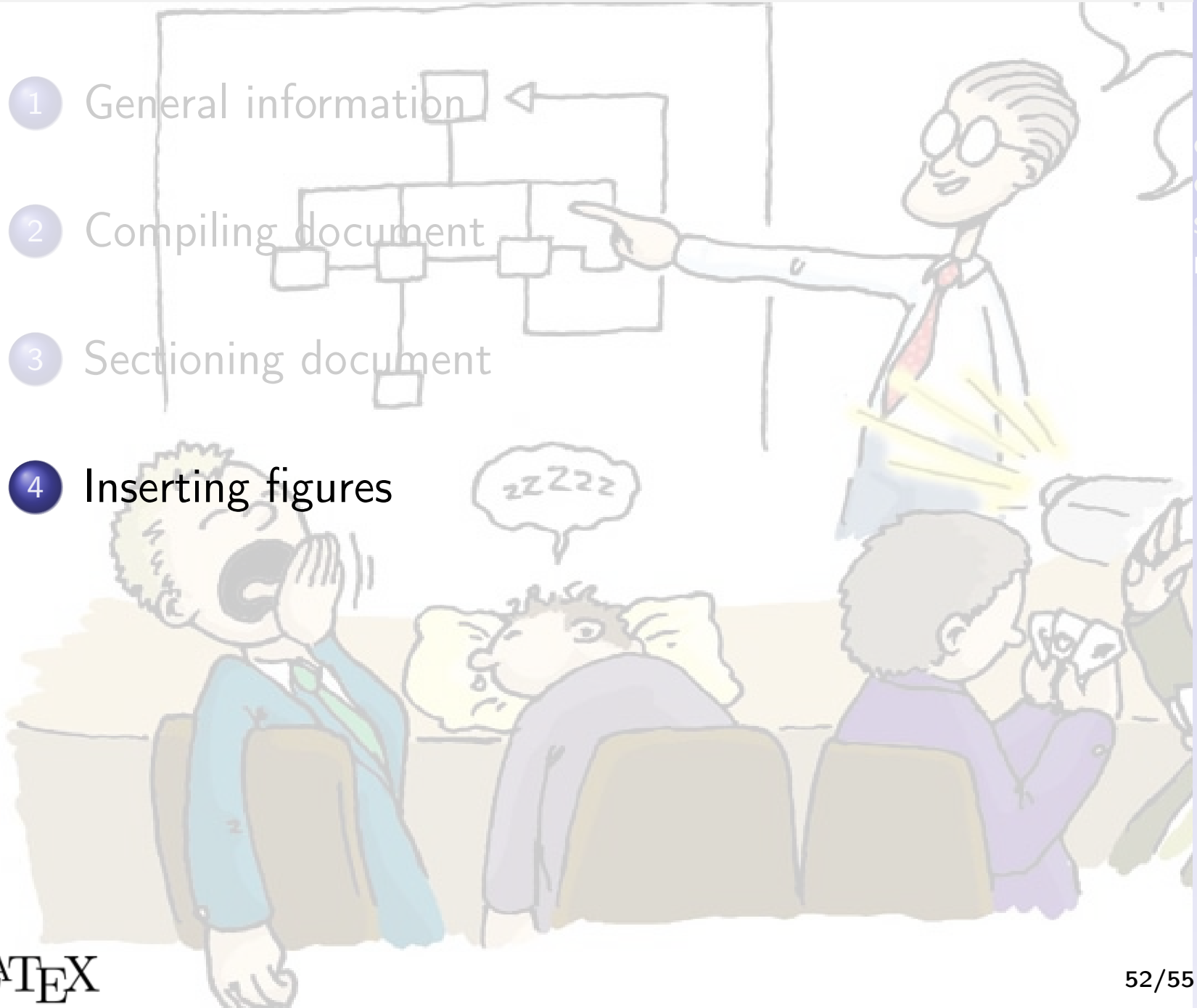
Usage:
\UseSecondLanguage{YES}
where the argument {} is not relevant, only if this macro is defined or not.
The two kinds of macros can be mixed, but only the 'D' macros are sensitive to changing the language.

In dual language documents, usually
`\MEDframe[keys]{subtitle, first language}{content, first language } {subtitle, second language} {content, second language}`
is used. I.e. the user provides titles and contents in both languages, and for preparing the output, selects one of them with `\UseSecondLanguage`.

Correspondingly, the biggest unit in a dual language document is the 'Dchapter'. (As mentioned, for slides it is transformed to 'Dsection'.) Usage:
`\MEDchapter[short title1]{long title1}{short title2}{long title2}`
which is transformed to
`\MEchapter[short title1]{long title1}` or `\MEchapter[short title2]{long title2}` calls, depending on whether `\UseSecondLanguage` is or is not defined.

The usage of the lower units is absolutely analogous.

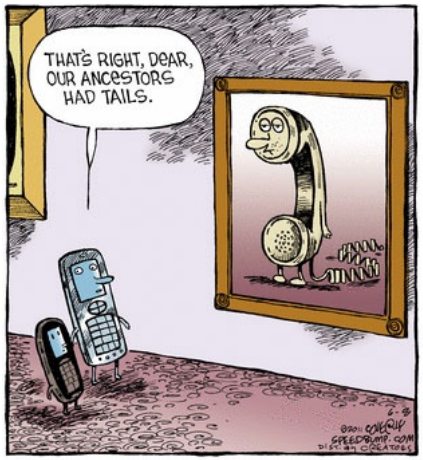
Some book styles also allow presenting some illustration at the beginning of the chapters.
Usage:
\MEchapterillustration{file}
For slides, the illustration appears in a 'plain' style style.
For books, the picture is placed at the beginning of the chapter. If the file name is empty, a 'fig/DefaultIllustration.png' file is searched. If the file not found, no illustration generated.
If macro \DisableChapterIllustration is defined, no picture generated.



4 Inserting figures
Traditional figures

Traditional figures can be displayed using macro `\MEfigure[keys]{image file} {caption} {label} {copyright} {ScaleFactor}`. Possible keys: `wide`.

©2011 http://pinterest.com



When new and old phones meet

On slides, the single-width figures are placed in 'columns'

The command used to display Figure was

```
\MEfigure{fig/phone_anchestors} {When new and old phones meet} {fig:phonenachestors} {2011 http://pinterest.com}{.8}
```


TEX
talk


TEX
talk