

From Mapping Files to Applications



www.safe.com



Peter Laulund

Product Developer, National Survey and Cadastre Denmark



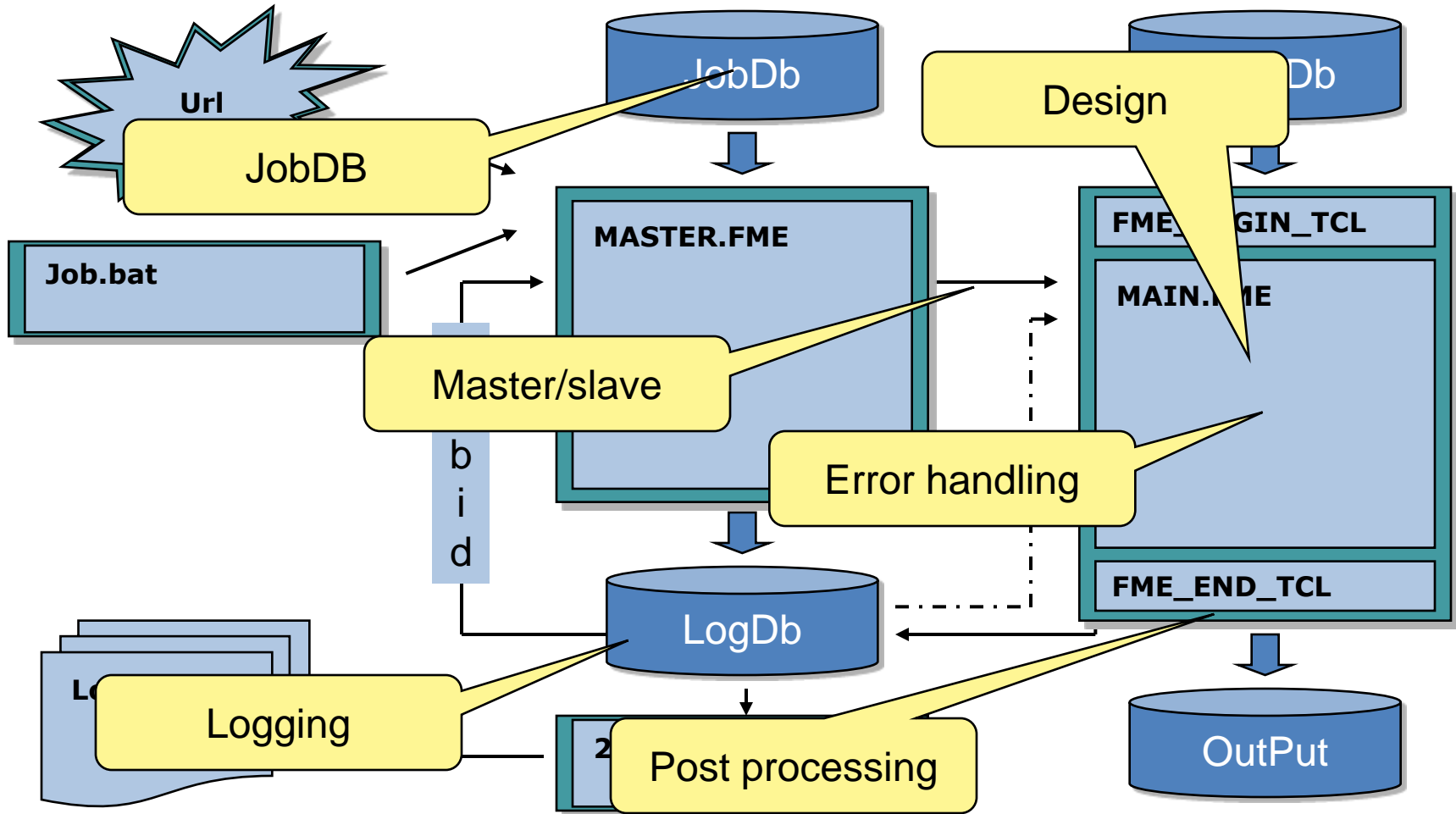
At the National Survey we work with:

- Several different datasets
- More than 15 different output formats and projections
- Complex query's
- Use the web and batch jobs to distribute data
- Some translations are don by non specialists
- Combining the translation with zip, email and ftp
- Log all jobs to a database

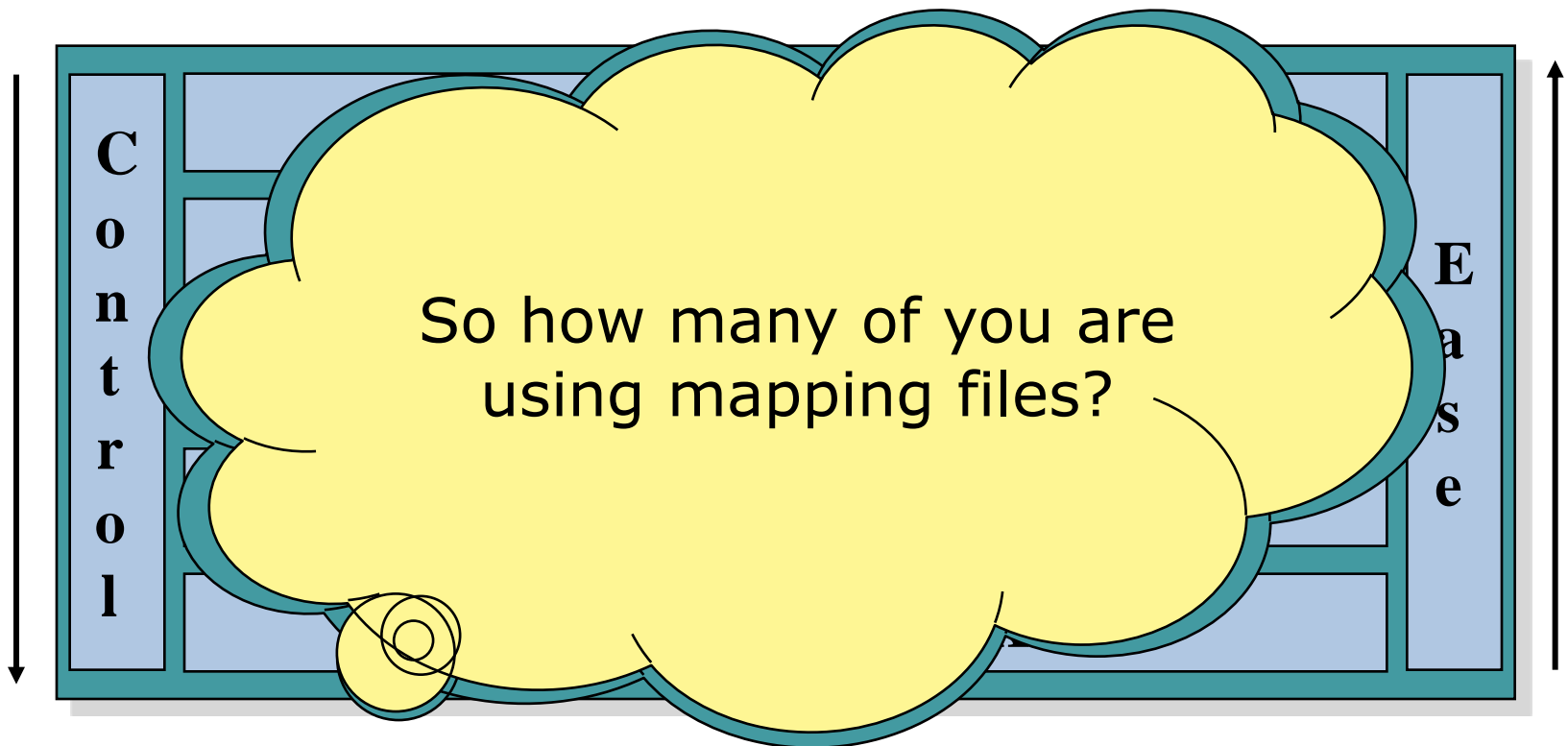
My challenge is to develop solutions that are:

- Flexible
- Easy to maintain
- Robust and able to handle errors
- Include pre and post processing
- My focus will be on the ideas I am working with more than on the technical implementation

FME Application – Just4Fun

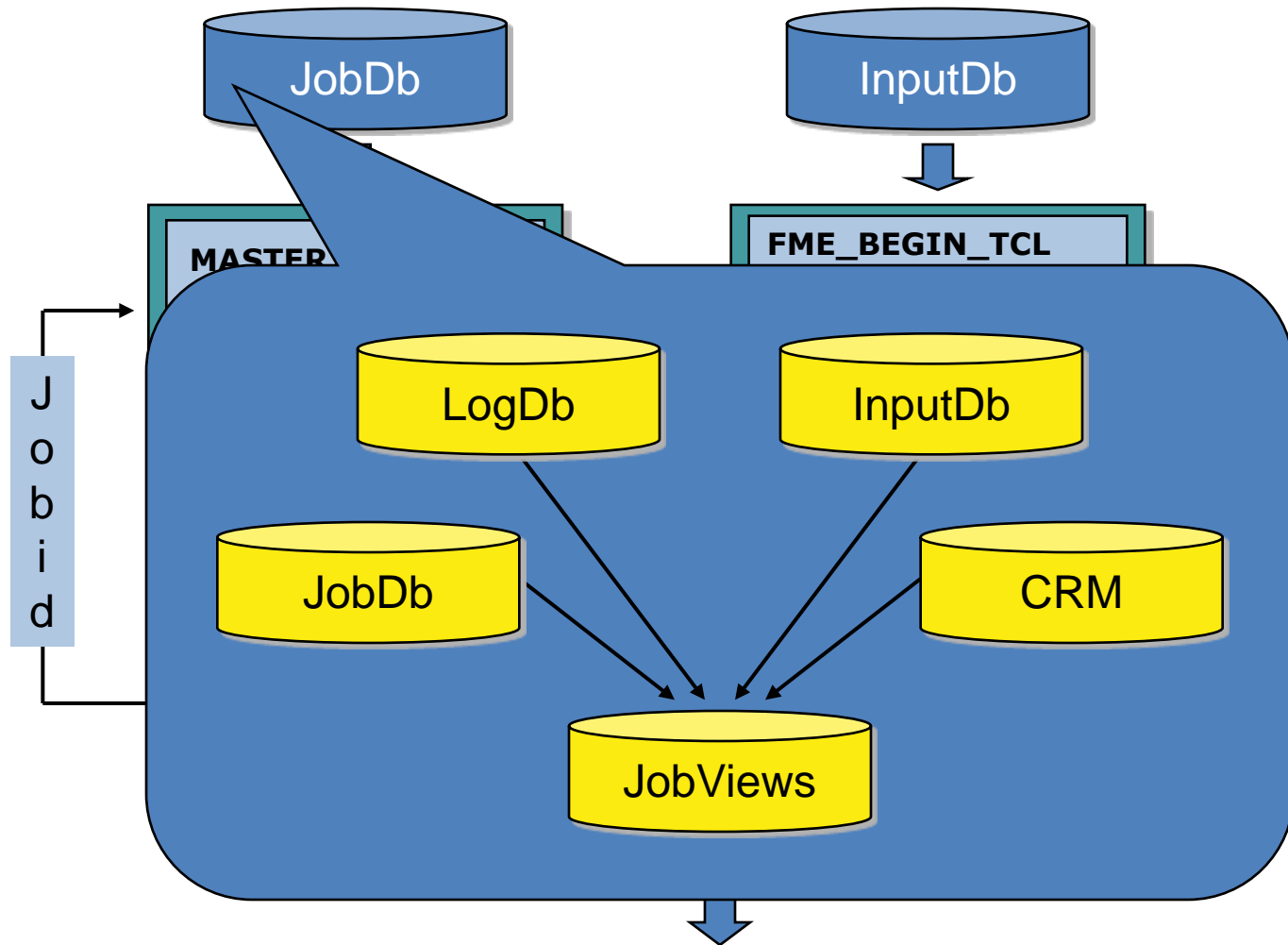


Mapping files versus FME Workbench is a matter of control

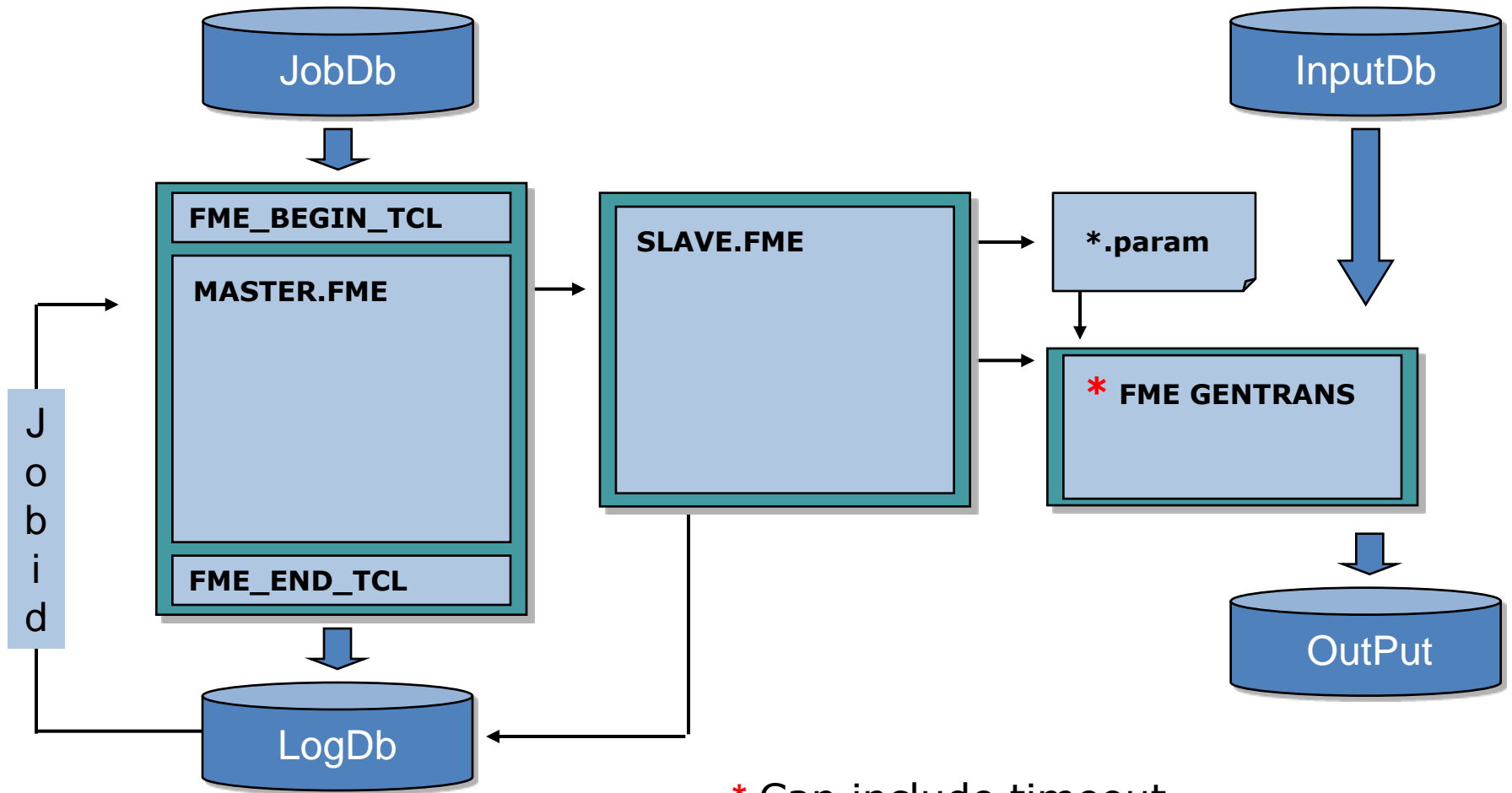


FME Application Architecture

www.safe.com



FME Application Architecture



* Can include timeout

Advantages of master/slave architecture:

- Can use FME toolbox to control FME
- Extra level of error handling
- Split a complex job into more simple sub processes
- Add looping and validation to input data resulting in a more flexible GUI
- Database driven batch jobs
 - Any input format - dBase, Oracle, csv, xls, ...

We can implement master/slave design with:

- Tcl and Python
 - With **fmeserverconsole.exe** we can use FME Server
- WorkspaceRunner transformer in FME Workbench
- ServerJobSubmissionFactory
- Can run in synchronous / asynchronous modes

```
lappend cmd fme.exe myMapFile.fme
lappend cmd --DestFormat $FME_MacroValues(DestFormat)
lappend cmd --DestCoordSys $FME_MacroValues(DestFormat)
lappend cmd --AdmKode $FME_MacroValues(DestFormat)
lappend cmd --Dataset $FME_MacroValues(DestFormat)

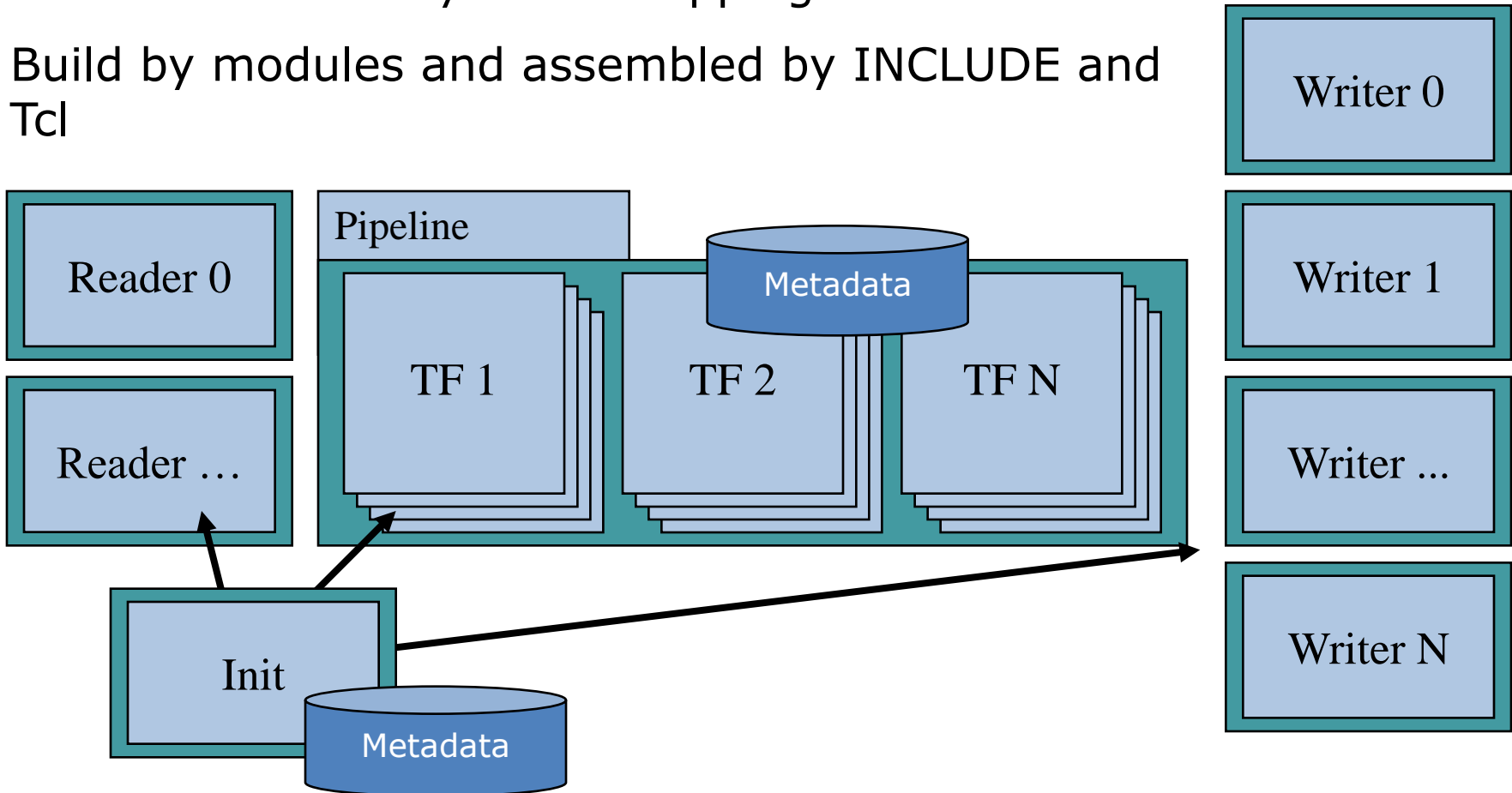
switch -- [ catch { eval exec $cmd 2> NUL: } msg ] {
    0 { FME_LogMessage fme_inform "MIA BATCH OK: " }
    default { FME_LogMessage fme_error "MIA BATCH FEJL: $msg"}
}
```

- A dynamic mapping file is a mapping file that changes as a function of the input parameters
- Advantages are:
 - Greater flexibility
 - Generic and reusable code
 - Better performance compared to FME Workbench
- Key elements in dynamic mapping files:
 - Normalizing data into a output neutral format
 - Standardization of output between formats
 - Schema mapping using external metadata tables

Dynamic Mapping File

Architecture of a dynamic mapping file

Build by modules and assembled by INCLUDE and Tcl



Dynamic Mapping File - Code

www.safe.com



```
# -----
INCLUDE [ set gsPath2App $(FME_MF_DIR_UNIX) ; \
          source $(FME_MF_DIR_UNIX)/TCL/lib/dbflib2.TCL ; \
          source $(FME_MF_DIR_UNIX)/tcl/include/getsql.tcl ; \
          source $(FME_MF_DIR_UNIX)/TCL/include/INITMAIN.TCL ; ]

INCLUDE $(mfRoot)\IMPORTERS\$(SourceFormat).FMI

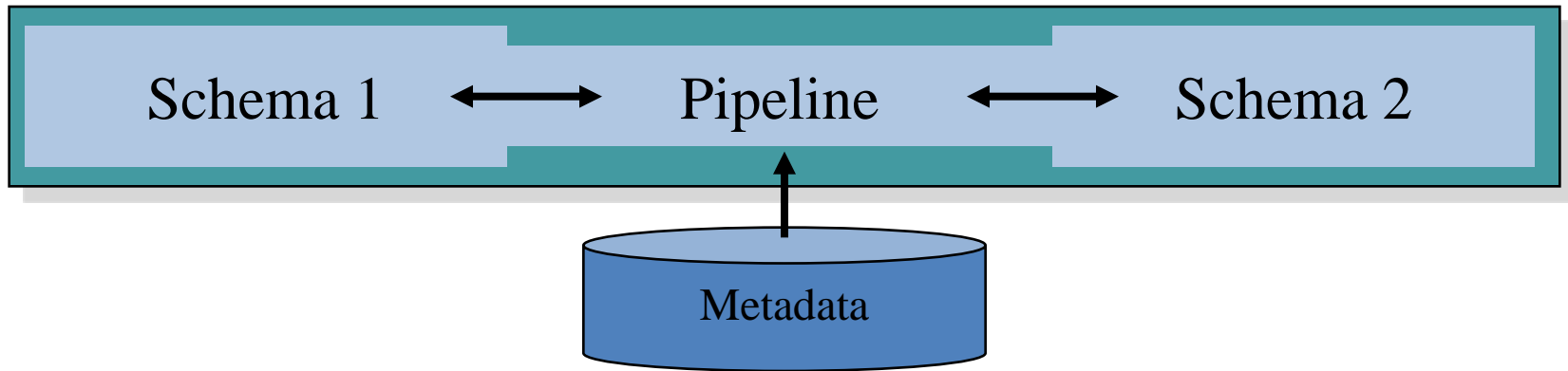
INCLUDE [ init::valAdmKode {$(AdmKode)} $(udtrType) ; \
          init::mail ; \
          init::dim $(Dataset) $(DestFormat) ; \
          init::getDefType $(DestFormat) ; \
          init::udtrType {$(AdmKode)} ]

INCLUDE [ init::cut $(udtrType) ]

INCLUDE $(mfRoot)/INIT/getJobId.fmi

INCLUDE [ init::ldsMain $(DestFormat) $(Dataset) {$(AdmKode)} \
          $(DestCoordSys) $(BBOX) $(X1) $(Y2) $(applikation) ; ]

INCLUDE [ init::getBoundingBox {$(AdmKode)} $(BBOX) $(udtrType) ; ]
```



In the **thin pipeline** we move all schema and attribute mapping from the pipeline into external metadata tables. All the pipeline does is handle the logic, so there is no explicit mapping in pipeline.

This way we get a simple mapping file and just have to update the metadata tables to maintain the application.

- There are no error free datasets!
- There are no error free scripts!
- We will therefore have to develop mechanisms in our script that will handle errors
 - Send an email to the administrator in the case of an error
 - Add default values to attributes
 - Test attributes and geometry and log all errors to the log file and FFS files

- Parsing errors – only mapping files
- Logical errors
 - Result in wrong output
- Environment errors
 - Missing connection to a database etc.
- Data errors
 - Geometry
 - Wrong geometry type, topological errors
 - Attributes
 - Missing attribute, wrong value

Data Errors – Example #1

www.safe.com



- Here we use tcl to add a default value to an attribute before we use it in a function call. We can use this approach with all type of function calls and calculations, where we are using attributes in calculations.

```
proc kmsOffset {} {  
    set aVal [ FME_GetAttribute dxy ]  
    set dxy [ expr {[regexp -- {[0-9]+(\.[0-9]+)?} $aVal ] : $aVal ? 0.0 } ]  
    FME_Execute Offset $dxy $dxy  
}
```

- Here we use a TestFactory to test that @Relate() was a success. If not, we tag the feature and then write it to an FFS file

```
FACTORY_DEF * TestFactory \
FACTORY_NAME "MINIMAKS INPUT: Juster attributter" \
INPUT FEATURE_TYPE * \
  @Relate($(Dataset), Read) \
TEST @Value(kms_join_ok) == yes \
OUTPUT PASSED FEATURE_TYPE * \
OUTPUT FAILED FEATURE_TYPE * \
  @SupplyAttributes(kms_rolle, kms_error ) \
  @SupplyAttributes(kms_error_kode, "ERROR: 1000" ) \
  @SupplyAttributes(kms_error_type, "ERROR: NoRelate") \
  @FeatureType(kms_error)
```

Why use a database rather than log files?

- Log files are great for development and debugging but too big for batch jobs
- Documentation
 - Internally – statistics
 - Externally – when did a customer get data
- Helps error tracking
- Give an overview of the applications state

- We log
 - All input macro-value pairs
 - Different kind of statistics – FME variables
- Implementation
 - We use an Oracle database, but any will do
 - In the master all features are written to the database
 - In the slave we use the oratcl library
 - In the database we set up a number of views presenting different statistics
 - With FME we can export the views to html

[<Example>](#)

- When we are using FME in batch mode or on a server we will often need to do some kind of post processing of the result as jobs run independently of any user.

Pre and Post Processing - Examples

www.safe.com



- Pre-processing
 - Deleting old data
 - Creating output directories
 - Copying documentation to output
- Post-processing
 - Zipping output data
 - Emailing to user / administrator
 - Logging to database
 - Moving result to ftp server

- FME Server pre and post processing is moved into the mapping file using FME_BEGIN/END_TCL and tcl scripts
- Result:
 - We get a more flexible solution
 - Can include logging and error handling
 - Will run even using FME Desktop

- In my presentation I have talked about the ideas I use to design my projects when we move from simple mapping files into more complicated applications using FMEServer and batch processing.
- I have focused on the concepts more than the actual implementation, as every example could fill a whole presentation.
- All the examples are implemented in running applications at the National Survey.

- So ***It's not All about Data*** but also a lot about Structure, Architecture, Design, Logging, Error handling and Post processing.

Thank You!

www.safe.com



- Questions?
- For more information
 - Peter Lauland: pelau@kms.dk & on FME Talk
 - Company name: www.kms.dk
 - Previous presentations:
www.fmeuc.com/2008 & www.fmeuc.com/2006
 - Tcl: wiki.tcl.tk &
www.tcl.tk/man/tcl8.5/TclCmd/contents.htm