MATLAB/Simulink use in Conception/Validation @ RENAULT
Overview

• Our activity

• MATLAB/Simulink and Renault

• Use of MATLAB to develop industrial applications

• Renault’s tools around MATLAB

• Examples

• Conclusion
Conception/Validation’s activity

Information Systems and Technologies Division
  – Computing development Department

• Our clients: Research and Design engineers in:
  – vehicle division
  – power train division

• Our job:
  – Analyze our clients’ process
  – Develop software (quality/cost/schedule)
  – Deployment
  – Maintenance
  – Support
Conception/Validation’s activity

- Scientific domains concerned (CAD/CAM/CAE)
  - Vehicle dynamics
  - Fluid dynamics
  - Acoustics
  - Vibration
  - Durability
  - Exhaust line and de-pollution modeling
  - Fuel efficiency and performance
  - Engine tuning and calibration
  - Hydraulic systems
  - …
Conception/Validation’s activity

- 70 in-house applications, 70 commercial software
  - 4000 users
  - Technologies
    - Languages: C++, C#, Fortran, Java
    - Software based: MATLAB/Simulink, Adams, I-DEAS, Fluent, Catia
  - \(\text{Nb lines of code} > 5 \text{ M}\)
- Our constraints
  - Reliability
  - Cost
  - Schedule
  - Worldwide deployment, ergonomics, multi-platform (UNIX, PC), flexibility, …
The choice of MATLAB/Simulink

• Productivity: minimize production cost and delivery time
  – Numerous functions embedded
    • GUI
    • Visualization tools
    • Simulink
    • Optimization
  – Multi-platform
  – High level language + matrix computation
  – Compiler

• Reliable

• Shared with engineering teams
  – good reactivity, close to client’s process
• MATLAB’s history at Renault
  – 1992 Introduction of MATLAB V4.2
  – 1996 First in-house industrial application
  – 1999 Upgrade to MATLAB V5
  – 2002 Upgrade to MATLAB V6
  – 2004 Upgrade to MATLAB V6.5.1
  – 2006 Upgrade to MATLAB V7 ???

• 2700 utilisateurs
• 30 MATLAB applications and numerous engineer models
• 30 MATLAB applications
  – LOC MATLAB > 500 000
  – 3 applications > 50 000 LOC MATLAB
  – Many “small” applications (~15 000 LOC)
• 3 Simulink applications
• 10 compiled applications
• 2 main libraries
  – OUTLAB (10 000 LOC MATLAB + 100 000 LOC C/C++)
  – OUPREDIM (7 000 LOC MATLAB + 80 000 LOC C/C++/Fortran)
MATLAB Run-time is available through a required library
• Development norms
  – Development scheme
  – General development norms
  – Architecture norms
  – MATLAB Development norms
    • 4 domains (General, GUI, Naming, Language)
    • 41 programming rules easily checkable

• Tools to increase productivity
  – XEmacs configuration
  – Code documentation tools
  – Code analyze tools
  – Compilation tools

• Re-usable components shared via libraries
Libraries

OUPREDIM  
- Read/Write files in specialized format  
- GUI included

OUTLAB  
- Links to C/C++/Fortran Libraries  
- Provides GUI components

OUEF  
- Container FE  
- C++

OID  
- Read/write files: UNV, Nastran, etc  
- C++

OUFIC  
- Read/write CSV files  
- C++

OUGZ  
- Compress / Uncompress files on the fly  
- C

- Evolves continuously (versions selected through library path)
- Contains  
  - Re-usable functions (read/write files)  
  - Correct dysfunctions / restrictions of MATLAB  
  - Common process
Multi-language

- Worldwide deployment
  - Multi-language applications

ENGLISH

Language Key

String Key

ERROR_READING_FILE

OUTLAB

Language Files

Translated String

‘Error while reading file’

ERROR_READING_FILE=‘Error while reading file’
Matxref: Documentation tool

- Documentation generation
- Cross reference and dead code analysis
• Audit tool
  – checks development norms
  – Generates a quality report
Matalone: compilation tool

- Builds a MATLAB project
  - One command to create an exe or a dll
  - Multi-platform
  - Outside MATLAB
  - Scriptable and generates a compiler report

- Quality control
  - Absent or erroneous code is detected

*** MATALONE.PL: Création de stand-alone Matlab

USAGE: matalone.pl -name:<name> [-main:<entry_func>] [options] project_dir

- main:<entry_func> désigne <entry_func> comme la fonction principale.
- name:<name> nomme le programme créé <name> ou lib<name> so (sur PC-Win: <name>.exe ou <name>.dll)
- no-compile ne fait pas la compilation (C->O)
- no-install ne crée pas le script de lancement
- no-generate ne fait pas la génération de M->C (défaut sous Windows)
- use:<file> utilise l’adaptateur <file> pour les composants externes.
Le fichier du répertoire courant est prioritaire sur celui proposé par Matalone
- verbose mode verbeux.

ATTENTION :
1/ Les prototypes des mex-fonctions doivent se trouver dans un repertoire src_matlab_prototypes_mex pour ne pas etre compile

2/ Les fichiers lang dans data/multiLangue sont compiles

Listes des adaptateurs disponibles par defaut:

- OUTLAB
- OUPREDIM
- LOGICALC
- OUTLAB_CAND
• Object modeling ensures the quality and upgradeability of a project
• MATLAB’s object oriented code is too restricted

• Object oriented development is achieved through a defined translation from UML to MATLAB code
  – Rules to implement
    • Class
    • abstract class
    • Interface
    • association (by value or reference)
    • Etc
Examples
• MATLAB represents a gain of productivity

• An industrial process for CAE applications around MATLAB is feasible and done at Renault

• Possible improvements
  – Quality analyzing tools
  – Test tools
  – Multi-language
  – Requirement management
  – OO aspects
Any question?