Safety-critical systems design: the TASTE tool-chain
High-integrity software constraints

- Real-Time determinism
- Safety & security
- Memory & processing constraints
Usual development process: myth

1. Specifications by designers
2. Validation by engineer
3. Development by voodoo coders
4. Tests, verification by engineers
5. Release by business consultants/sales dept.
Usual development process, overview

- Design
- Specifications
- Validation
- Validated specifications
- Program (binary)
- Implementation
- Verification, qualification
Usual development process, reality (1)
Usual development process, reality (2)
Funny but ...

- Nor for life-/mission- critical systems
- Must do the *dirty* and *boring* work
- And do it *correctly*
In addition ...

- Requirements and constraints increase
  - Number of functions and their impacts
  - Costs (money, time)

- Allocated resources decrease
  - Budget
  - Time, release to market

- Cannot use traditional methods
Key points

- Validation
- Automation
- Verification
Ideal development process

1. Specifications by designers

2. Validation by *engineer* analysis tools

3. Development by *voodoo-coders* code generators

4. Tests/verif by *engineers* execution analysis tools

5. Release by business consultants/sales dept.
TASTE guidelines

• Abstract software & hardware

• Focus on engineering concerns

• Validate & verify as early as possible

• Automate as much as possible
TASTE process

1. Define **system interfaces**

2. Abstract **soft & hard aspects**

3. Validate & verify **requirements**

4. Generate application using ACG
TASTE development process

Design

Specifications

Validation

Validated specifications

Verification, qualification

Program (binary)

Implementation
TASTE benefits

Traditional process

TASTE process
TASTE workflow

Specifications
- Interfaces specifications
- Software models
- Deployment models

Verification & Qualification
- System execution
- Documentation generation
- Run-time analysis
- Software metrics acquisition

Validation
- Scheduling
- Trade-off analysis
- ...

Automatic Code Generation
TASTE technologies (1)

1. System interfaces: **ASN.1**

2. Soft specifications: **C/Ada, Simulink, SDL**

3. Hard deployment & conf: **AADL**
TASTE technologies (2)

Specifications

- Cheddar
- MAST
- Ocarina/REAL

Validation

- ASN1 Compilers
- Ocarina
- Matlab/Simulink
- ...

Automatic Code Generation

- COUVERTURE
- Qemu
- Gprof
- GNUplot

Verification
TASTE use-case

ARM movement acquisition

Data transmission through PCI

ARM movement reproduction

Data transmission through ethernet

TASTE system

- Data acquisition from devices
- Heterogeneous software (Simulink, RTDS, bare-C)
Demonstration

1. Interfaces and functions specifications
2. System validation
3. Automatic implementation
4. Verification
Conclusion

✔ OSS tool-chain for safety-critical systems

✔ Support by industry & academia

✔ Evaluation with real developments
Perspectives

- Enrich validation aspects
- Design OSS application code generators
- Improve verification tools
http://www.assert-project.net/taste