POK - Introduction

Julien Delange <julien@gunnm.org>
Forewords

• The POK project
  • Design and implement safe and secure system
  • Complete development process with model-based engineering

• POK O/S
  • Part of the POK project
  • Can be used independently
  • Support several standards (ARINC653/POSIX/...)

Julien Delange <julien@gunnm.org> - http://pok.gunnm.org
Outline

- POK rationale and overview
- POK objectives and guidelines
- Functionalities
- Platforms
Outline

• POK rationale and overview

• POK objectives and guidelines

• Functionalities

• Platforms
Rationale - problems

- Safety-critical systems are difficult to build
  - Many requirements
  - Dedicated architectures
  - Specific standards: ARINC653, MILS, DO178B, ECSS, etc

- Need to validate/verify/specify/certify
  - At specification
  - At execution
  - Require an appropriate development process
Rationale – solution overview

• Framework to build safety-critical systems
  • From specifications down to the implementation
  • Enforce requirements at each development step

• Help designers as more as possible
  • But do not force him!
  • POK keeps you free!
Solution – Architecture outline

• Partitioned architecture
  • Time and space isolation
  • Resources isolation across partitions

• Fixed amount of resources
  • Avoid run-time overhead
  • Improve determinism
Solution – Architecture outline – cont'd

Partition 1 (size = 5Mo)
- 3 tasks
- 2 intrapart channels
- 1 out interpart port

Partition 2 (size = 4Mo)
- 4 tasks
- 1 intrapart channel
- 1 in interpart port

Isolation/Partitioning Kernel
- Isolation partitions in memory segments
- Allocate 500 ms for each partition
- Monitor inter-partitions communication
Architecture outline – space partitioning

- Segmentation isolation
  - Determinism
  - Sometimes, emulate segmentation (e.g. LEON port)
- Allocate segments at initialization time

Organization of memory
Architecture outline – time partitioning

- Strict time isolation
- Round-Robin policy for partition scheduling
- Partition-dependent scheduling algorithm
Technology overview (1)

- Modeling framework (AADL modeling)
  - AADL modeling patterns
  - Requirements validation through AADL analysis
- Partitioned operating system & code generation
  - Provide time and space partitioning
  - Ensure requirements enforcement
- Certification tools
  - Verify implementation against specification requirements
  - Certify implementation against certification standards
Technology overview – cont’d

- Full development process
  - Some steps could be achieved manually
  - We don't force developers to do good things

Spécifications

1. Validation
2. Implementation
3. Certification
Relation with current work

- We propose a nice development process
  - Requirements validation and enforcement
  - Many tools that may be well developed by computer geeks

- But we want industrial solutions!
  - Standard compliance
  - Ex: ARINC653, MILS, etc...
Standards integration

ARINC653 or MILS modeling patterns

ARINC653 or MILS requirements validation

1. Validation

2. Implementation

3. Certification

Spécifications

ARINC653 automatic code generation

DO178B automatic certification
Outline

- POK rationale and overview
- POK objectives and guidelines
- Functionalities
- Platforms
POK objectives

• Build safe and secure systems
  • Provide a complete framework for the whole dev process
  • Detect errors as early as possible
  • Assist the user in each development step

• Potential industrial impact
  • Integrate industry-proven approaches
  • Development process can be adapted to customer needs
POK guidelines

• KISS !
  • Very simple: must be easy to read/understand/ maintain ...
  • … but very, very stupid

• Keep the window open !
  • Potential kernel certification
  • Openness to other projects (ex : COUVERTURE)

• Highly-customizable toolchain
  • You can throw away some tools
  • Integration of other tools (ex : OSATE, Eclipse, ...)
POK functionnalities

• Specification-level
  • System consistency, safety & security validation
  • Automatic code generation for kernel & partitions

• Runtime level (POK O/S functionnalities)
  • Partitioning support
  • Standards supports : ARINC653, POSIX

• Certification level
  • Code coverage analysis of both kernel and partitions
  • Execution instrumentation
POK architecture support

- X86 intel
  - Generic architecture
- SPARC
  - LEON3 board
  - Aerospace domain
- PowerPC
  - Wide-used architecture
  - Embedded-purpose domain
Questions ?