Using Ada's Visibility Rules and Static Analysis to Enforce Segregation of Safety Critical Components

J-C. Van-Den-Hende
ALSTOM Transport
jean-christophe.van-den-hende@transport.alstom.com

J-P. Rosen
ADALOG
rosen@adalog.fr

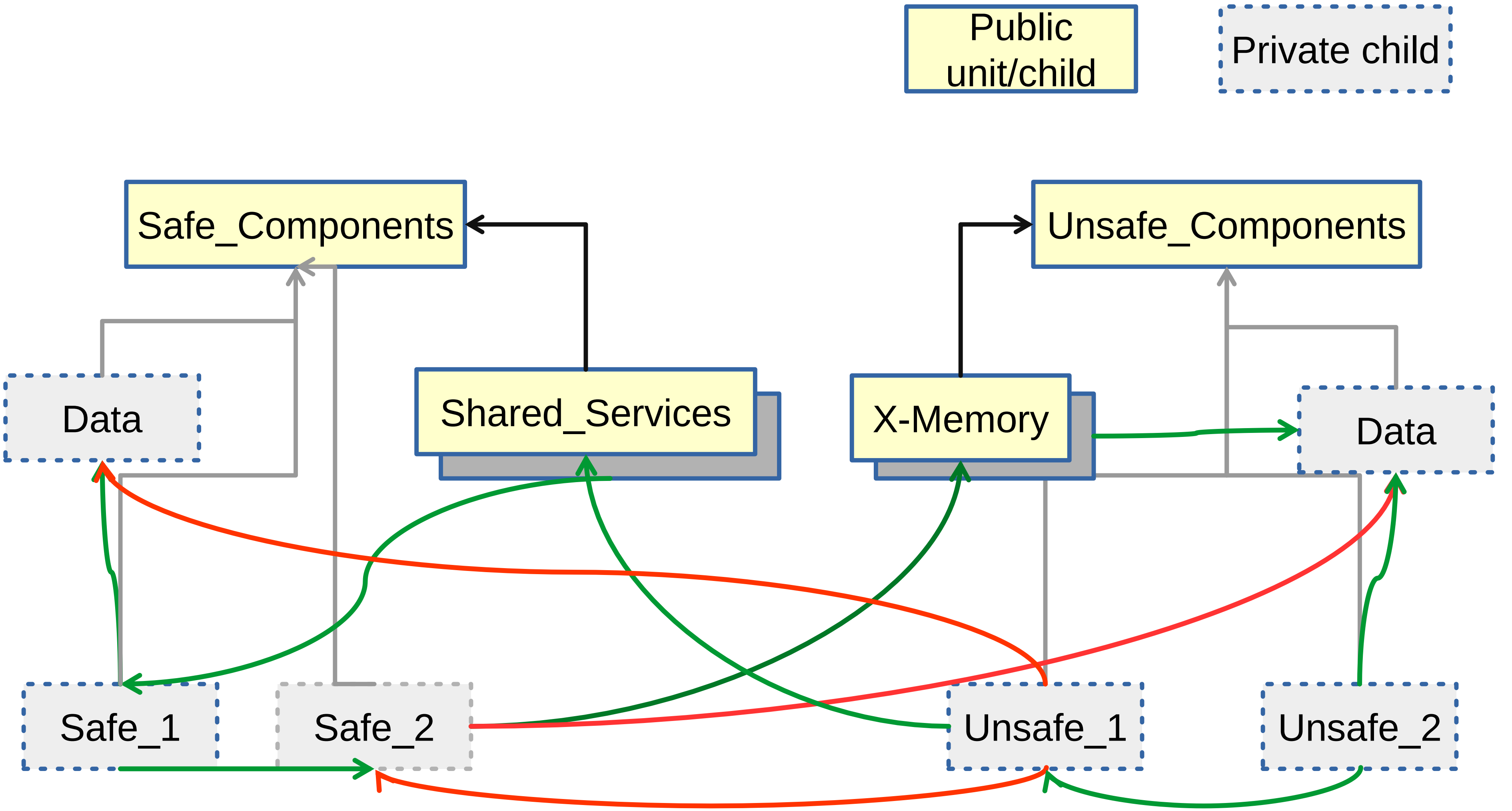
Safety Integrity Levels and Segregation

- Railway systems: EN50128 defines 5 “integrity levels”
  - From SIL0 (not critical) to SIL4 (highest criticality)
  - Similar to DO178B/C levels **reverse** A ..E
  - Constraints (and costs!) increase with SIL level
- Mixed criticality:
  - Same computer running various criticality applications
  - Same application with various criticality components
  - How to make sure that unsafe components do not alter safe ones?
- Possible solutions
  - Validate all components at highest level (expensive!)
  - Hardware protection
  - Proofs

Alstom Segregation Requirements

- Components based architecture with only two levels: SIL0 (not certified) and SIL4 (certified) components
- Data can be passed from SIL0 to SIL4
  - ☞ Deemed unreliable
- ✓ SIL4 access must go through special gateways to check validity
  - ☞ No direct access of SIL4 data by SIL0 components
- Some components are not by themselves SIL4, but may be called by SIL0 as well as SIL4 components
  - ☞ Classified as SIL4
- SIL4 components shall call SIL0 components only through special isolation components
- SIL0 components shall not call other SIL4 components

Structure


Public unit/child
Private child
Safe_Components
Unsafe_Components
Data
Shared_Services
X-Memory
Data
Safe_1
Safe_2
Unsafe_1
Unsafe_2

Other Checks

- ×No unchecked programming
  - ☞Verified by AdaControl

- ×No removal of language checks, including in SIL0 components
  - ☞Verified by AdaControl

- ×No visible variable in package specifications
  - ☞Verified by AdaControl

Achievements

- Criticality of a component is immediately identifiable from its full name
  - ☞The name defines applicable rules
  - ☞Cross-criticality accessors are easily identified
- The most important rules of segregation are enforced by proper usage of language features
  - ☞Violations don't compile!
- Remaining rules are checkable by static analysis

Name another language that can achieve that...