Real-time languages

- An informal scan of the real-time (embedded, dedicated, safety-critical) market:
  - 30% assembly and legacy languages
  - 30% Ada
  - 30% C/C++
  - 10% other (100+ other languages)
- C and Ada are the most commonly used languages in civil aviation today
- C++ is gaining popularity, but its usage is still

Kornecki, A.J.; Zalewski, J.; , "Software Development for Real-Time Safety-Critical Applications," Software Engineering Workshop - Tutorial Notes, 2005. 29th Annual IEEE/NASA , vol., no., pp. 1- 95, 3 & 8 April 2005 doi: 10.1109/SEW.2005.6

Feature	RT-ADA	C/C++	JAVA
Memory management	Automatic	Manual	Garbage collection
Run-time Efficiency	High	High	Medium
Run-time Predictability	High	OS-dependent	Low
Concurrency controls	Language features	OS specific	Language libraries

*Real-time languages

*Standards for RT OS

* RT-POSIX (Real-time standard for making UNIX programs portable)
* OSEK/VDX (automotive industry, standardization of communication infrastructure)
* ARINC 653
* Micro-ITRON

*RT-POSIX

* Portable Operating System Interface for UniX (POSIX) is a family of standards.
* RT-POSIX is the real-time extension, it provides:
  * system calls for concurrent programming
  * Task synchronization via mutual exclusion by priority inheritance
  * Task synchronization via condition variables,
  * Data sharing among tasks
  * Prioritized message queues for inter-task communication

*RT-POSIX

* Features:
  * Fixed priority preemptive scheduling,
  * Sporadic server scheduling
  * Time management with high resolution,
  * Sleep operations,
  * Multipurpose timers
  * Execution time budgeting
  * Virtual memory management

*OSEK/VDX

* OSEK/VDX is a standard of the automotive industry which addresses safety-critical real-time applications.
* Applications are grouped into functions which are mapped to a huge number of resource units.
* As main objective the standard defines the communication environment for automotive control unit applications.
* It also includes a standardization of the inter-networking interfaces for ensuring safety and reliability of communication networks.
* The standard does not achieve portability of the software, as the I/O interaction of the devices is not part of the standard.

*ARINC 653

* ARINC 653 (Avionic Application Standard Software Interface 653) for implementing, integrating and certifying analyzable safety-critical RT applications for integrated modular avionics architectures
* Main idea: **space and time partitioning**, which excluded interference and makes systems predictable.
* Each partition represents a different application, with its own memory space and its time slot for accessing a resource, e.g. processor

*ARINC 653

* Applications are made of tasks which communicate via message buffers, semaphores and events and executed under fixed priority static scheduling
* Tasks of different partitions communicate via exchanging messages over ports provided by the API of the underlying OS.
* For task communication it is transparent if the receiver is located on the same processor, a different partition or just an interface device
* Ports can be used in a sampling mode and a queuing mode. In the former mode not ready messages are overwritten

*Micro-ITRON

* Micro/ITRON (Industrial RT OS Nucleus) is a de-facto standard for embedded systems in Japan.
* Standardized kernel functions, called **standard profiles**
* The standard profiles support task priorities, semaphores, message queues and mutual exclusion primitives with priority celling and priority inheritance