TTP/A Konzept
OMG Standard

Hermann Kopetz
TU Wien

September 2001

Outline

- ♦ Technology Change
- ♦ Why do Computer Systems Fail?
- ♦ Design Faults
- ♦ Composability by an Architecture Approach
- ♦ TTP/A as a Member of the TTA
- ♦ Conclusions

Technology Change: Challenge or Crisis?

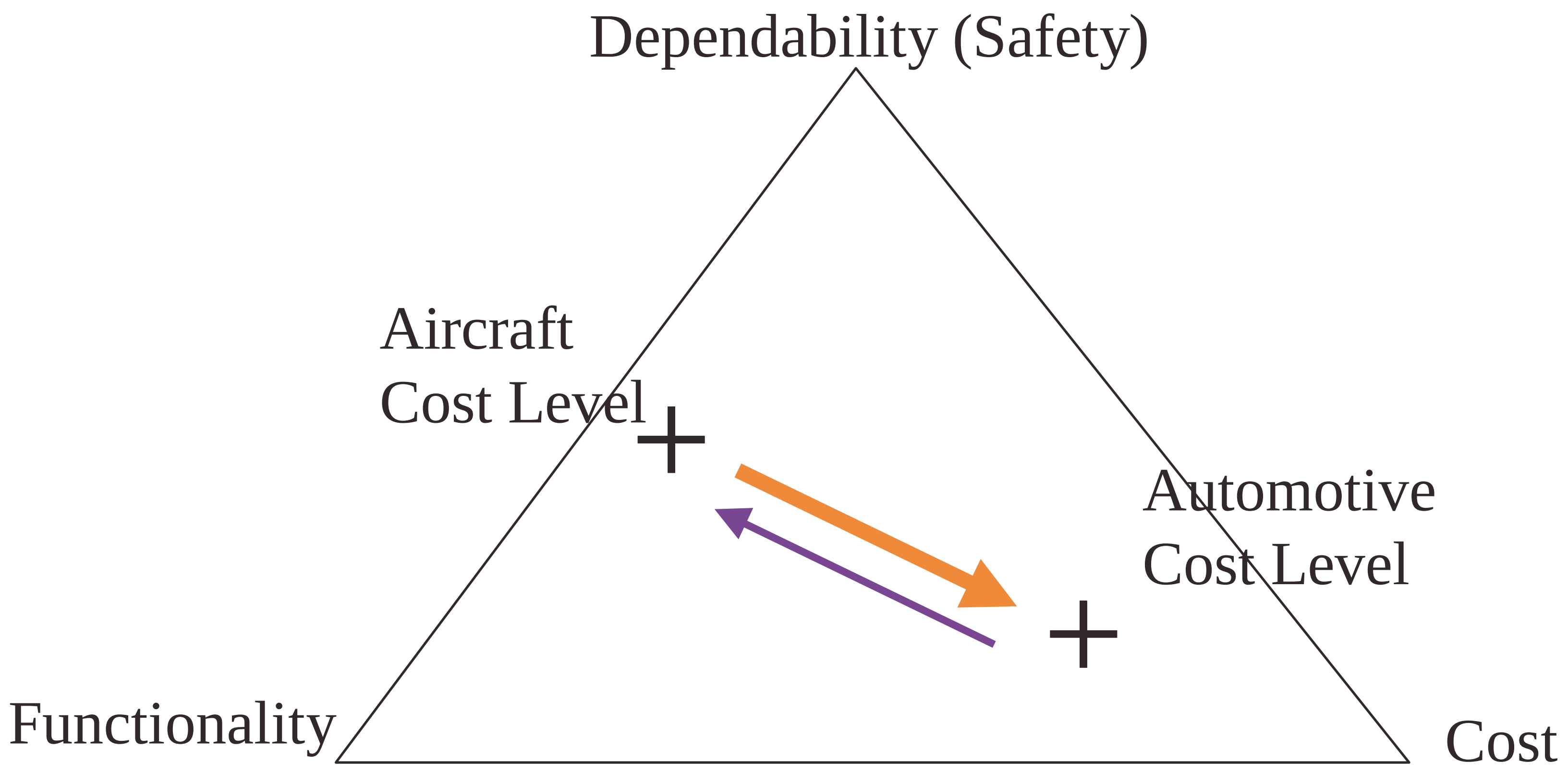
It is only a question of “**when**”, and not of “**if**” different industries will be forced to make the transition form mechanical/hydraulic control systems to computer-based control systems.

Tomorrow will not be like today.

The aircraft industry has managed this transition successfully in a high-dependability environment. This implies that the technologies to built high-dependability electronic systems are available.

The automotive industry will be next :
“Drive-by-Wire” will follow “Fly-by-Wire”

The Challenge Problem for the Automotive Industry



to bring safe systems to the automotive cost level.

What can you do Today with 1 mm^2 of Silicon

- Build a 32 bit wide processor (e.g., the ARM 7 processor)
- implement 100 k-bytes of memory (e.g., the 256 Mbit memory chip from Infineon is less than 100 mm^2).

Today, the marginal production cost (without IP, packaging,etc.) of 1 mm2 of silicon is in the order of 10 US cent.

Communication capabilities increase even faster than processing capabilities.

Moore's Law Lives

Intel announced technology that can shrink circuits even further- keeping the chip-speed rule on track through 2007, or even 2009.

At a conference in Kyoto, Japan, Intel displayed transistors, or circuits, only 70 to 80 atoms wide. This nanometer technology should lead to low-power chips containing 1 billion transistors running at speeds of 20 GHz. (Today's fastest Pentium 4 models have 42 million transistors and run at 1.7 GHz.)

The coup de grace: These feats can be accomplished using current chipmaking equipment, not with future innovations.

THE INDUSTRY STANDARD MAGAZINE, Mark Boslet, Date: Jun 25, 2001

Consequences of Moore's Law

- Hardware cost will be dominated more by the number of packages, then by the functionality of the silicon real-estate in each package.
- Separation of mixed signal chips (e.g, smart MEMS transducers) from large logical processing chips.
- The use of the smart sensor technology will increase.
- Distributed architectures are the only alternative.
- Because of the decreasing feature size, transient hardware faults will increase--need to provide fault-tolerance.

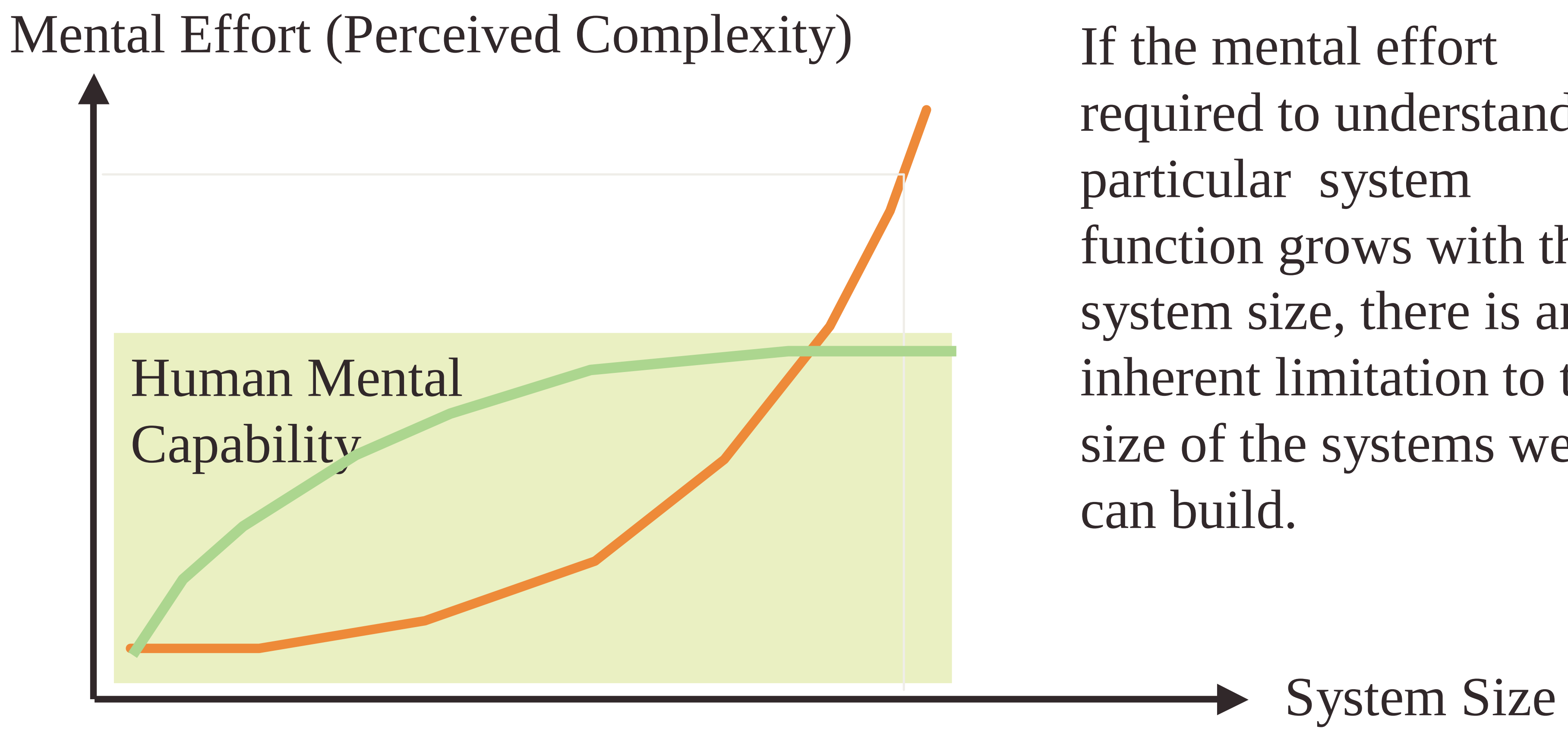
Why Do Computer Systems Fail?

- **Internal Physical Faults:** The cause of the failure is, e.g., a physical aging process within a chip. Can be transient (soft) or permanent. *It can be assumed, that multiple failures of chips are statistically independent*--will increase due to reduction of feature size.
- **External Physical Faults:** The cause of the failure is a disturbance external to the chip, e.g., EMC, spikes in the power supply, mechanical shock. Can be transient or permanent. *It **cannot** be assumed that multiple failures of chips are statistically independent.*
- **Design Faults (Software Faults):** The cause of the failure is the design (software or hardware) resulting in inconsistent states and actions. Different components of the same design will fail at the same instant.

What are the Mechanisms to Handle these Faults?

- ♦ **Internal Physical Faults:** Systematic or application specific redundancy, e.g., replication of components. Should not increase the complexity of the application software.
- ♦ **External Physical Faults:** High-quality engineering, replication is not the solution, since failures are statistically dependent.
- ♦ **Design Faults (Software Faults):** Reduction of the Complexity of a Design--making a design understandable.

Design Faults: Reduction of Complexity



If the mental effort required to understand a particular system function grows with the system size, there is an inherent limitation to the size of the systems we can build.

Design faults have their root in unmanaged complexity.

How to Reduce Complexity?

- ♦ Partition a system into *nearly autonomous* subsystems.
- ♦ Support *composability* at the level of the architecture such that subsystems can be developed and validated independently
- ♦ Define an *architecture style* to avoid *property mismatches* at the interfaces.
- ♦ Provide *well-defined (in time and value) and small* interfaces between the subsystems, supported by a *two-level design methodology*.
- ♦ Transfer *generic system services* (e.g., clock synchronization, membership, fault-tolerance management) into the hardware.

Composability--The Problem

Given a set of components, e.g., smart transducer nodes, each one of it provides a *prior service* across a linking service interface (LIF).

How do we have to design these LIFS in order to be able to determine a priori whether a given composition of the set of components will provide the intended *emerging service*?

Examples for Property Mismatches

Property	Example
Physical, Electrical	Line interface, plugs, 12 V versus 42 V power net
Communication protocol	CAN versus J 1850
Syntactic	Structure of the data, Endianness of data
Flow control	Implicit or explicit, Information push or pull
Incoherence in naming	Same name for different entities
Data representation	Different styles for data representation
Temporal	Different time bases or inconsistent time-outs
Dependability	Different failure mode assumptions
Semantics	Differences in the meaning of the data

How to handle Property Mismatches?

- The *architecture style* sets common standards that eliminate some of the “lower level” property mismatches!
- Agreed syntax of interface data items and operations, e.g., expressed in IDL (Interface Definition Language of CORBA)
- Precisely specified temporal properties of the interfaces
- Agreement on the “meaning” of the data items

The Principles of Temporal Composability

A composable architecture must avoid property mismatches by enforcing a common *architectural style* on all components and must support the

- *Independent development of components*--relates to the architecture
- *Stability of prior services*--relates to the components
- *Constructive integration of components*--relates to the communication system.
- *Replica determinism*--to support transparent implementation of fault tolerance.

The TTA supports these principles of composability.

Technical System Architecture

- An architecture is a framework for the construction of a system out of subsystems (components).
- Architectural style: The architecture must provide guidelines for the partitioning of a system into subsystems and for the design of the interactions among the subsystems.
- Components must be designed to comply with the *architectural style* to avoid packaging mismatch.
- An architecture must *constrain* an implementation in such a way that the ensuing system is understandable, maintainable, extensible, and can be built cost-effectively.

Vision of the Time-Triggered Architecture

Development of a generic architecture for high-dependability distributed real-time systems that can be applied in the many different application domains

- ♦ Automotive
- ♦ Aerospace
- ♦ Railways
- ♦ Industrial Control
- ♦ . . . .

Our vision comes closer to reality by the decision of Audi to use the TTA in the automotive domain, by Honeywell to use the TTA in aerospace domain, and by Alcatel to use the TTA in the railway domain.

Priorities in the TTA

- **Safety without compromises**
  - No single point of failure
  - Formal analysis of critical functions
- **Composability:**
  - Building systems out of prevalidated components--Platform electronics
  - Fully specified interfaces in the temporal domain and value domain
  - Two level design methodology
- **Flexibility**

and all of these considering the automotive cost constraints.

Design Principles of the TTA

- ♦ Provision of a consistent distributed computing base (Membership service)
- ♦ Unification of Interfaces
  - • Real-Time Service Interface (TT)
  - • Diagnostic and Management Interface (ET)
  - • Configuration Planning Interface (ET)
- ♦ Temporal Composability
- ♦ Transparent Fault-Tolerance
- ♦ Scalability and Openness

Two-Level Design Methodology in the TTA

- ♦ at the **System Integrator Level** the interactions between the subsystems are designed and the CNIs to the components are fully specified in *the value domain and in the temporal domain*.
- ♦ at the **Component Supplier Level** the components are designed and implemented, taking the precise CNI specifications as design constraints.
- ♦ Two-level design approach activity supported by a tool set from TTTech, the high-tech spinoff from the TU Vienna

The composability property of the architecture ensures that the component properties (e.g., timeliness) are not invalidated by the system integration.

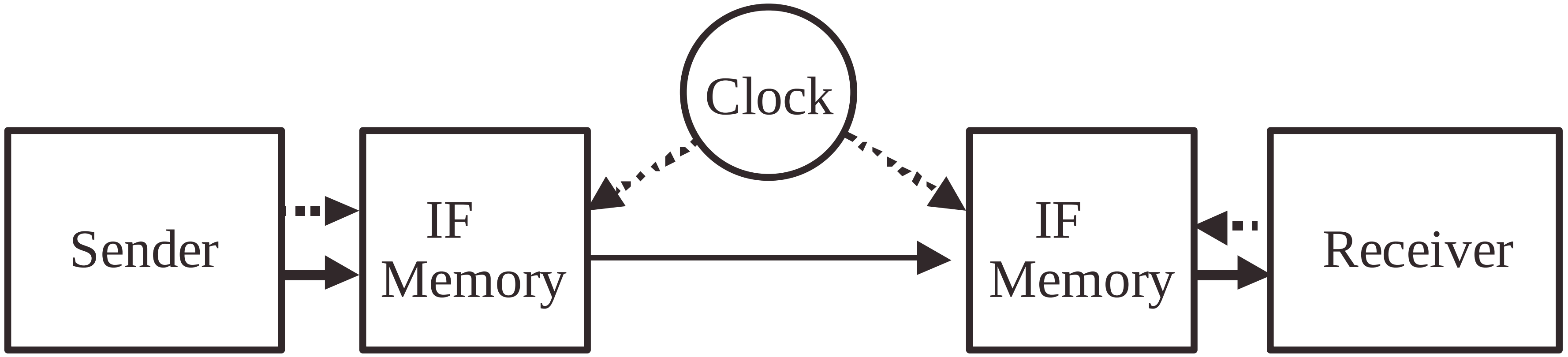
Architecture Design *is* Interface Design

A good interface within a real-time system

- is precisely specified in the value domain and in the time domain,
- provides the relevant abstractions of the interfacing subsystems and hides the irrelevant details,
- leads to minimal coupling between the interfacing subsystems,
- limits error propagation across the interface,

and thus introduces ***structure*** into an architecture.

Information Transfer in a TTA Interface

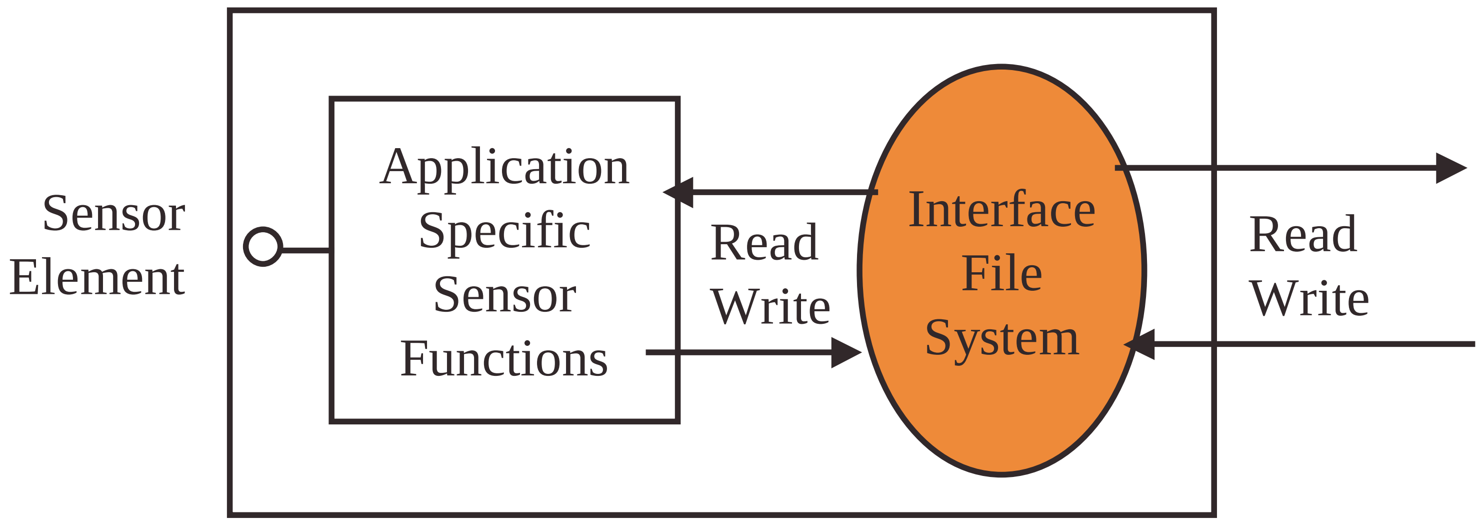


Information Push
Ideal for Sender

Time-Triggered
Communication System

Information Pull
Ideal for Receiver

Interface File System of the TTP/A



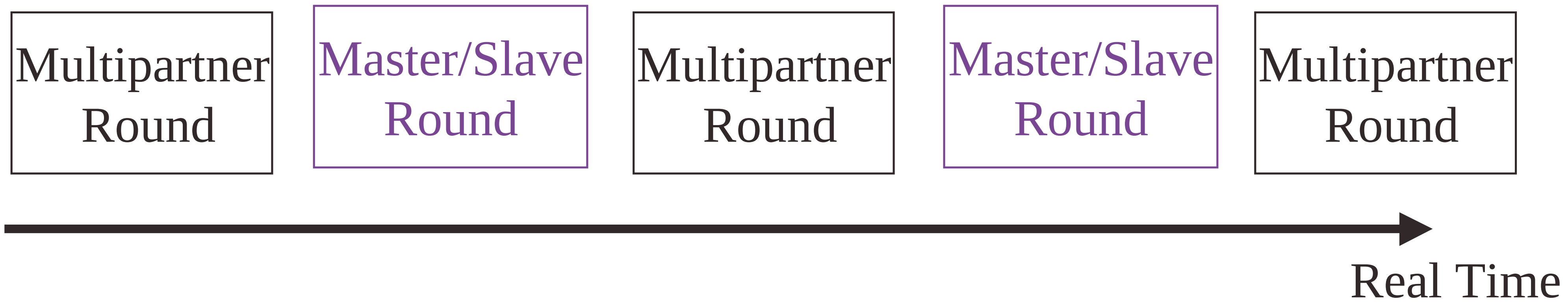
The Interface File System (ISF) encapsulates all information that is exchanged between a smart transducer and its environment.
It provides a standardized structured name-space for information access

TTP/A--Principle of Operation

- ♦ Endpoint of the communication is a record in an Interface File System (IFS) located in the transducer node.
- ♦ Communication is organized into Rounds
  - A round is started by the active master that has knowledge of the global time
  - The first frame of a round is a fireworks frame, followed by data frames. The structure of a round is described in the round-descriptor list (RODL).
  - every round is independent of every other round
- ♦ The arrival of the fireworks frame is the global synchronization event starting a new epoch.

Interleaving of Rounds

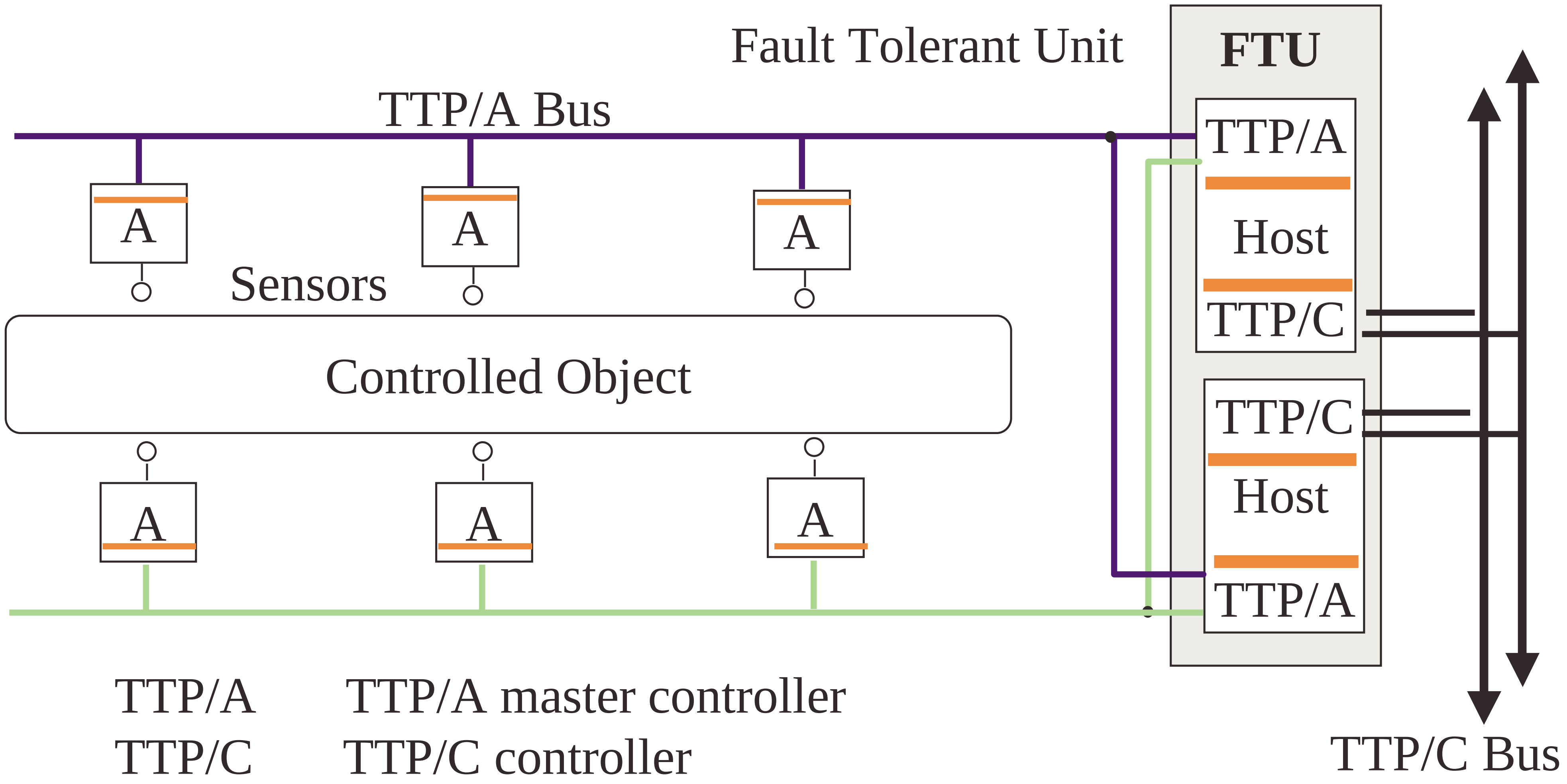
Recommended Schedule:



Master Slave Rounds have constant frame length.

Master Slave Rounds may be empty, if no CM or CP service is requested by the master.

Fault-Tolerant Sensor Connection



TTP/A	TTP/A master controller
TTP/C	TTP/C controller
A	TTP/A slave node interfacing to sensors and actuators

OMG History

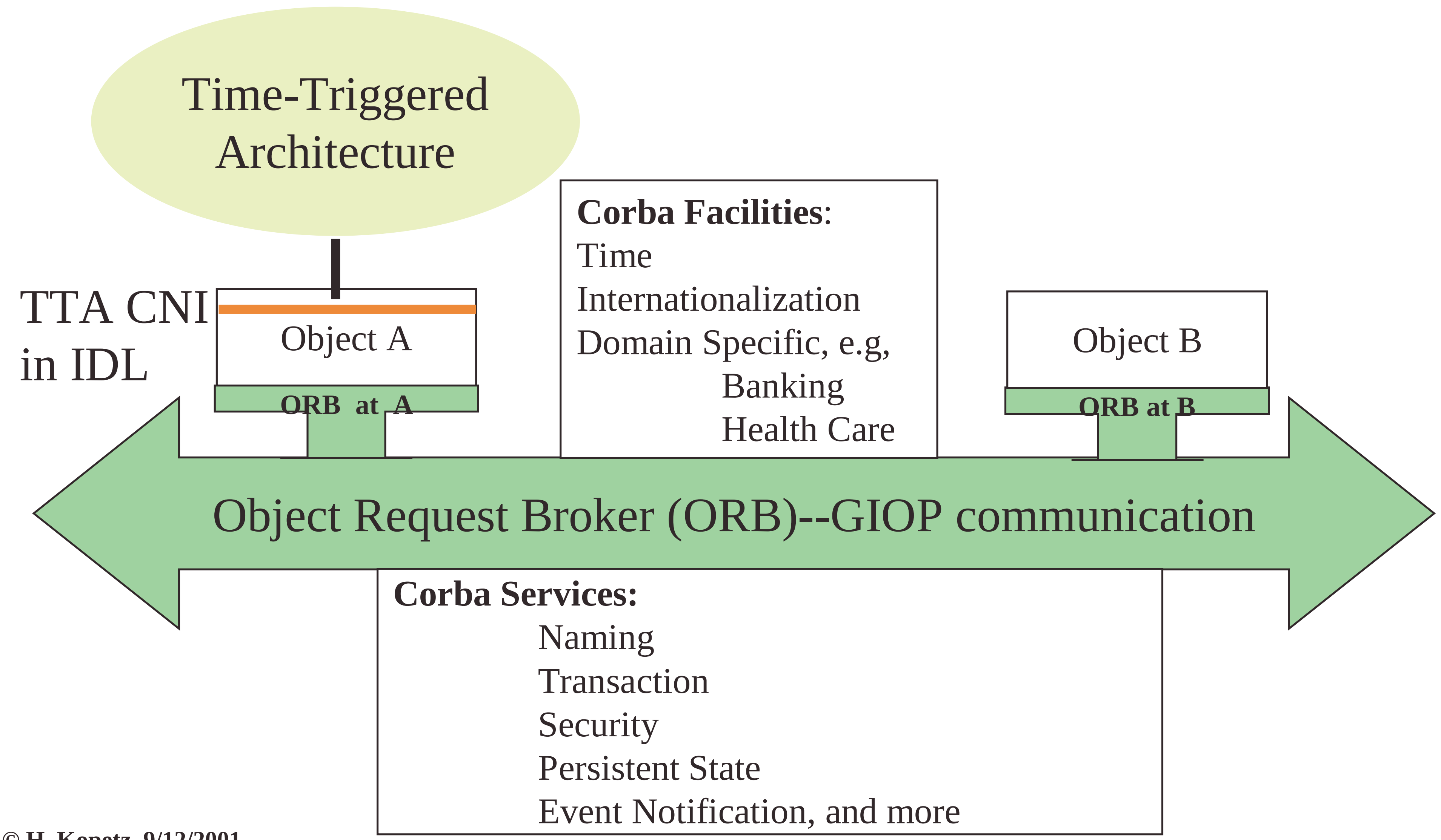
The Object Management Group (OMG) was established in 1989 with the mission to “provide a common architectural framework for object-oriented applications based on widely available interface specifications”.

At the beginning of 2001, the OMG has more than 800 members.

The OMG achieves its goal by the establishment of the Object Management Architecture (OMA).

At the core of the OMA is the Common Request Broker Architecture (CORBA).

TTA and the CORBA Architecture



Interface Definition Language (I DL)

The standardized OMG IDL is a notation that allows the programming-language independent specification of object interfaces.

- ♦ IDL is strongly typed
- ♦ Has an appearance that is similar to C++
- ♦ Supports the specification of data structures and operation names (methods)
- ♦ Supports the specification of exceptions.
- ♦ Can be compiled to different programming language

Summary: The TTP-A Protocol

- ♦ Universal Smart Transducer Interface
- ♦ Provides Standard Interface File System (IFS)
- ♦ Latency Guarantee for Control Applications, Clock Synchronization better than .1 msec
- ♦ Good Error Detection for fail safe operations
- ♦ Low Cost for intelligent sensors, smallest implementation less than 2 kbytes of ROM, 64 bytes of RAM (including IFS, software UART at 10 kbits on single wire)
- ♦ Fault tolerance at system level (duplicated buses)

In the process of standardisation by the OMG, the world's largest organisation for the creation of IT standards.

Architecture is the Key

"*As electronic system design priorities in the automotive industry migrate from hardware to software, architectural standardization and leadership become key success factors for the industry and its individual companies.*"

Quote from:

Electronics Architecture for the Millenium", AEI, July 2001, p. 111