REAL-TIME SOFTWARE

Dayang Norhayati Abang Jawawi

School of Computing
Faculty of Engineering
Universiti Teknologi Malaysia
email : dayang@utm.my

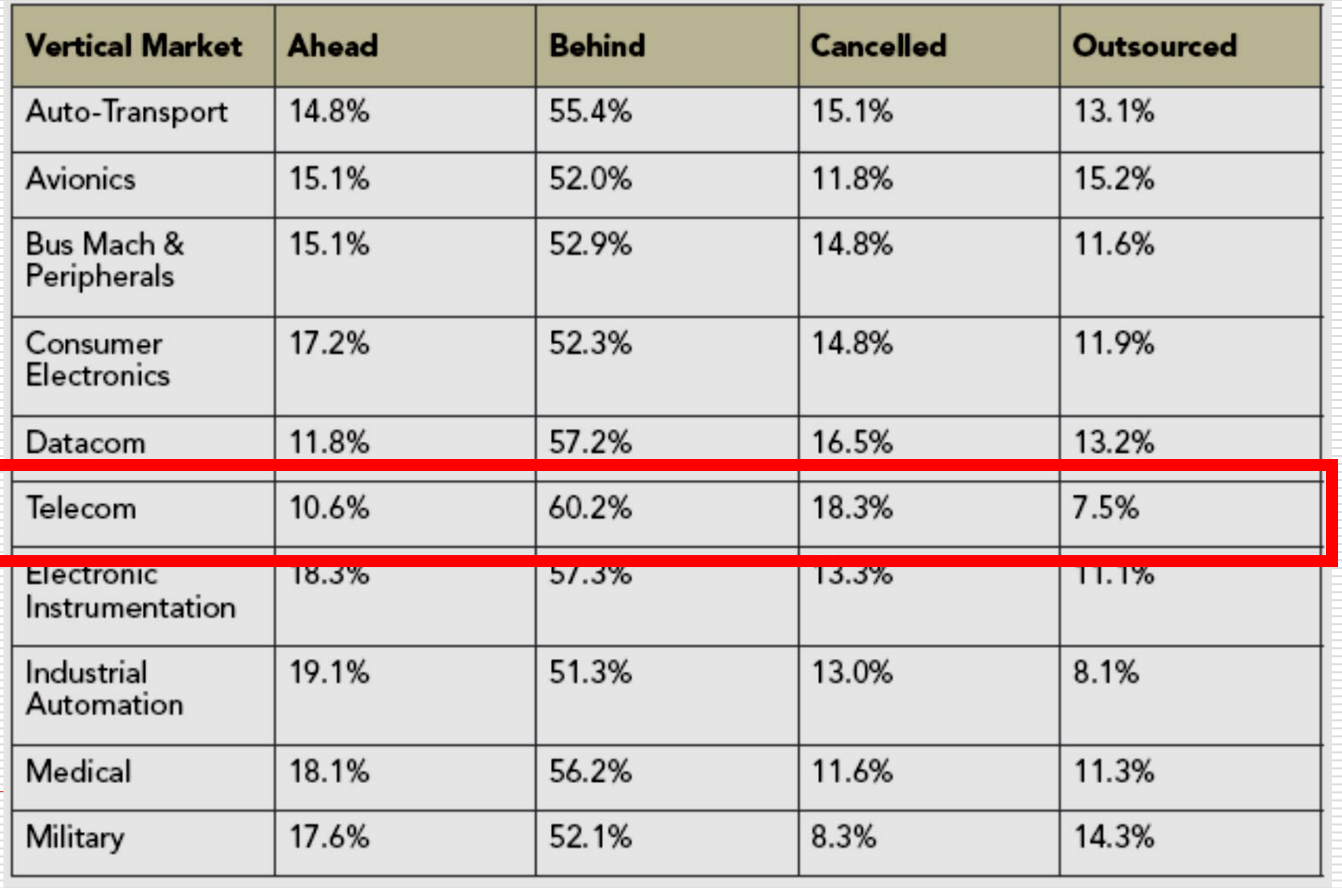
The Structure of Presentation

1. Challenges in Software Engineering of ERT Systems
2. Software Development Life Cycle
3. Software Engineering Essential for ERT
4. Conclusion

Challenges in Software Eng. of ERTS

Survey (2005) on embedded systems design completions – in all applications, > 51% are behind schedule!

Krasner J., "Optimizing Technology Choices For Device Software", *Embedded Market Forecasters,* May 2005



Vertical Market	Ahead	Behind	Cancelled	Outsourced
Auto-Transport	14.8%	55.4%	15.1%	13.1%
Avionics	15.1%	52.0%	11.8%	15.2%
Bus Mach & Peripherals	15.1%	52.9%	14.8%	11.6%
Consumer Electronics	17.2%	52.3%	14.8%	11.9%
Datacom	11.8%	57.2%	16.5%	13.2%
Telecom	10.6%	60.2%	18.3%	7.5%
Electronic Instrumentation	18.3%	57.3%	13.3%	11.1%
Industrial Automation	19.1%	51.3%	13.0%	8.1%
Medical	18.1%	56.2%	11.6%	11.3%
Military	17.6%	52.1%	8.3%	14.3%

Challenges in Software Eng. of ERTS

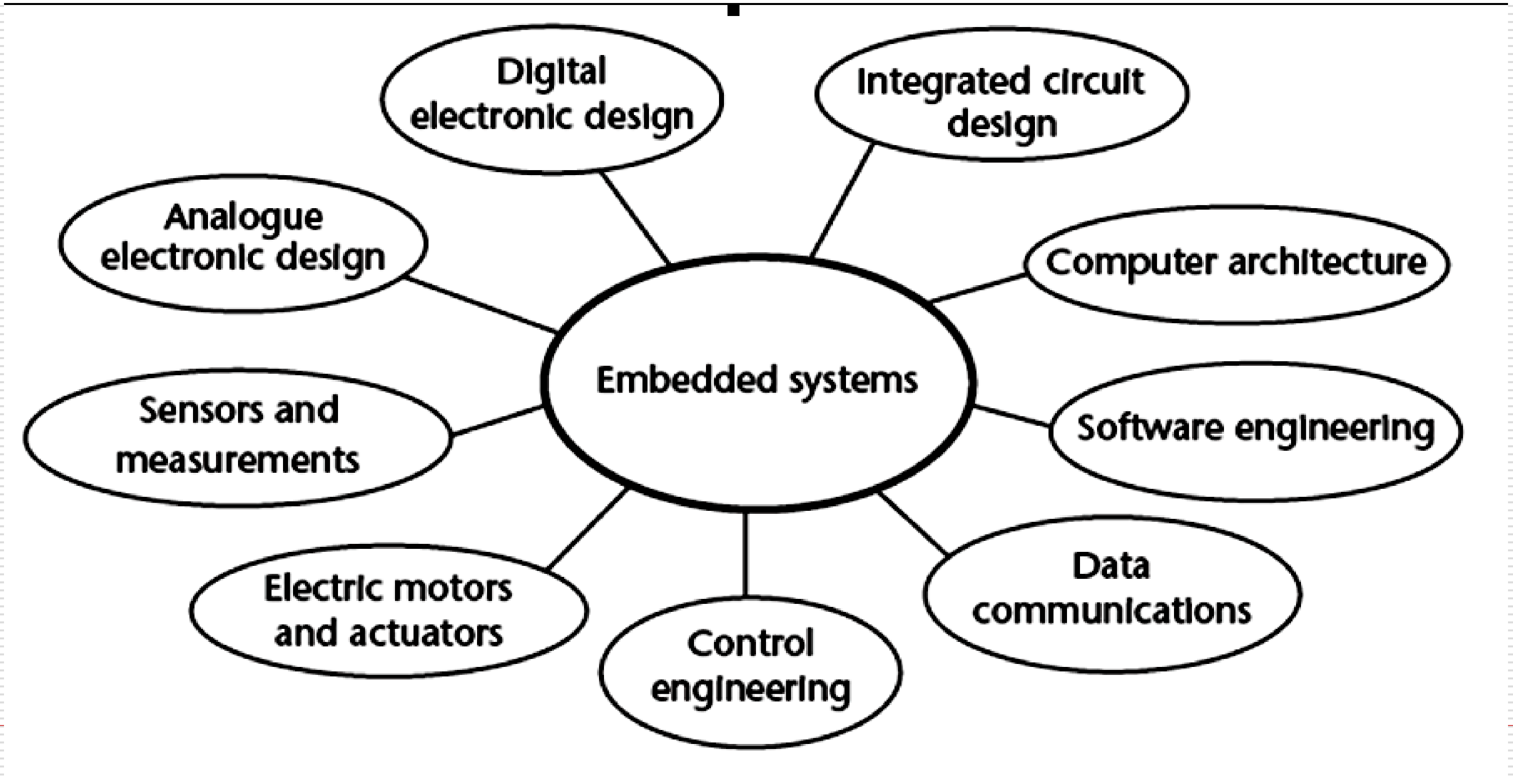
Many ERTS have limited resources such as memory, processing power, I/O etc. – Resource Constrained ERTS.

Resource Constraints can be due to requirements imposed on ERTS:

1. Low Cost (e.g in high-volume products) efficient use of h/w & s/w dev. budget are required.
2. Low Weight or small size products due to customer preference requires h/w implementation with single-chip MCU usually *with limited on-chip resources*.
3. Battery operated ERTS are expected to have long run-times, thus requires efficient energy usage – *clock speed should be as low as possible & only necessary h/w parts should be used.*

Challenges in Software Eng. of ERTS

Many ERTS have complex life-cycle & require multi-disciplinary skills.



Challenges in Software Eng. of ERTS

Some challenges to ERTS software engineers:

1. Software is becoming large & complex.

Large means a team of s/w engineers is required to complete the work in time.

Complex means not many s/w engineers have the multi-disciplinary skill required in ERTS.

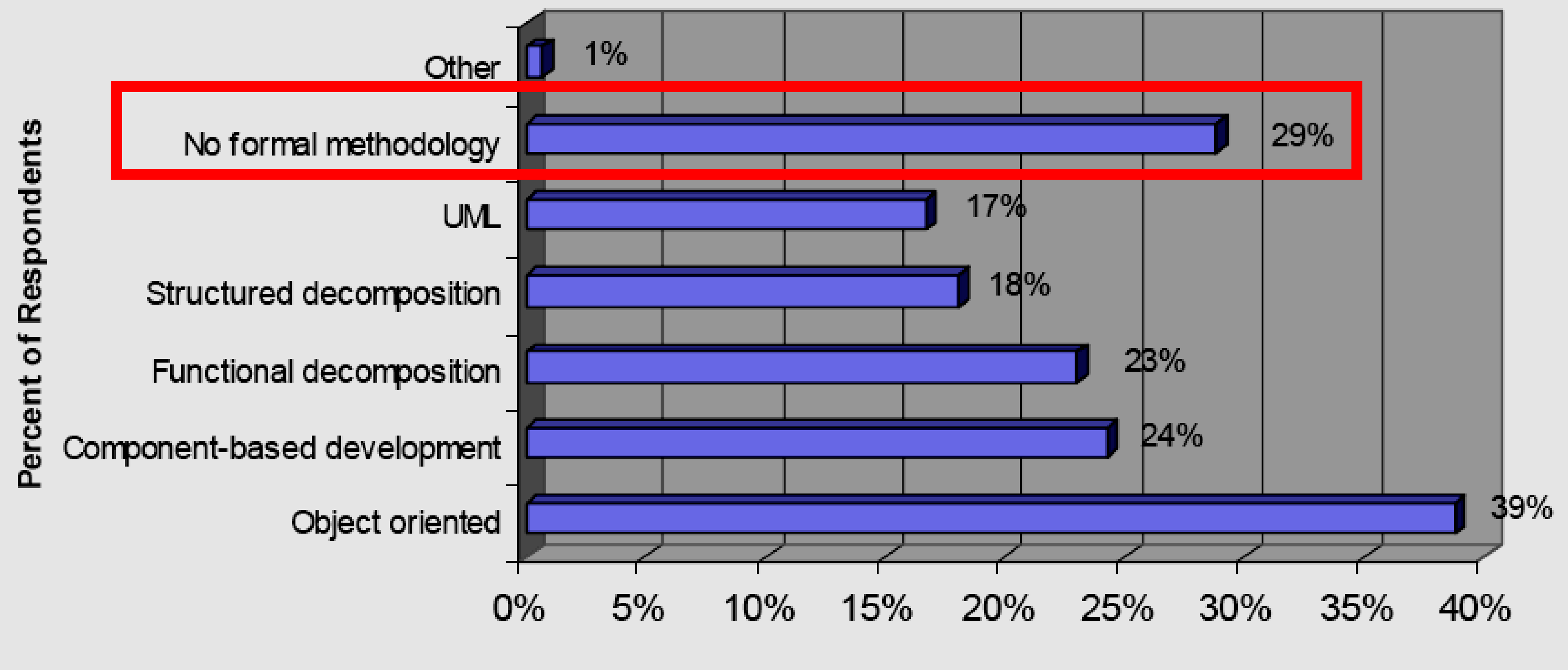
2. Software has to be created faster, better, cheaper, & more reliable.

3. "One size fits all" technology no longer works.

These are good incentives to systematically reuse code, design, architecture, component etc.

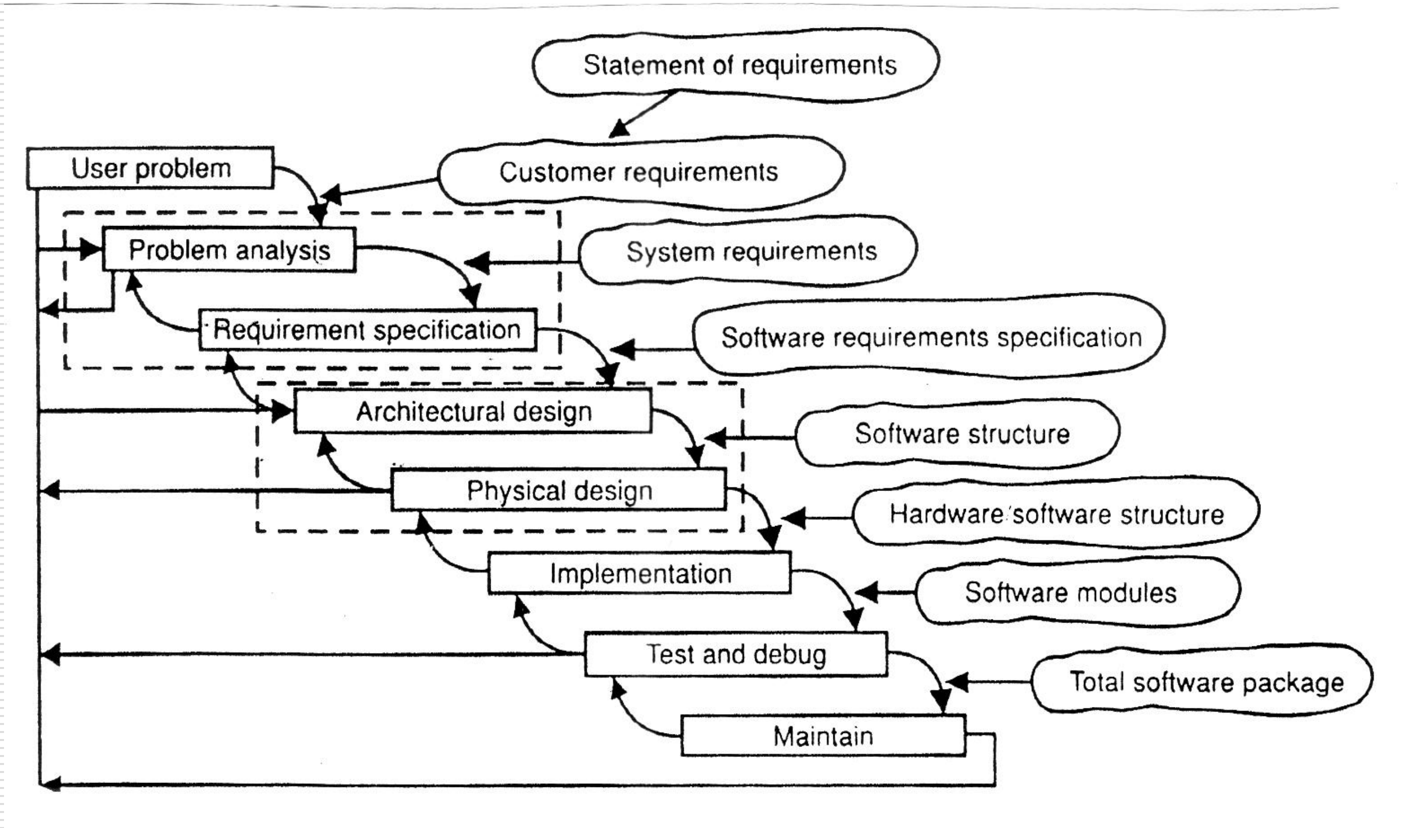
Challenges in Software Eng. of ERTS

Survey (2004) on methodologies used to design ERT software ❖



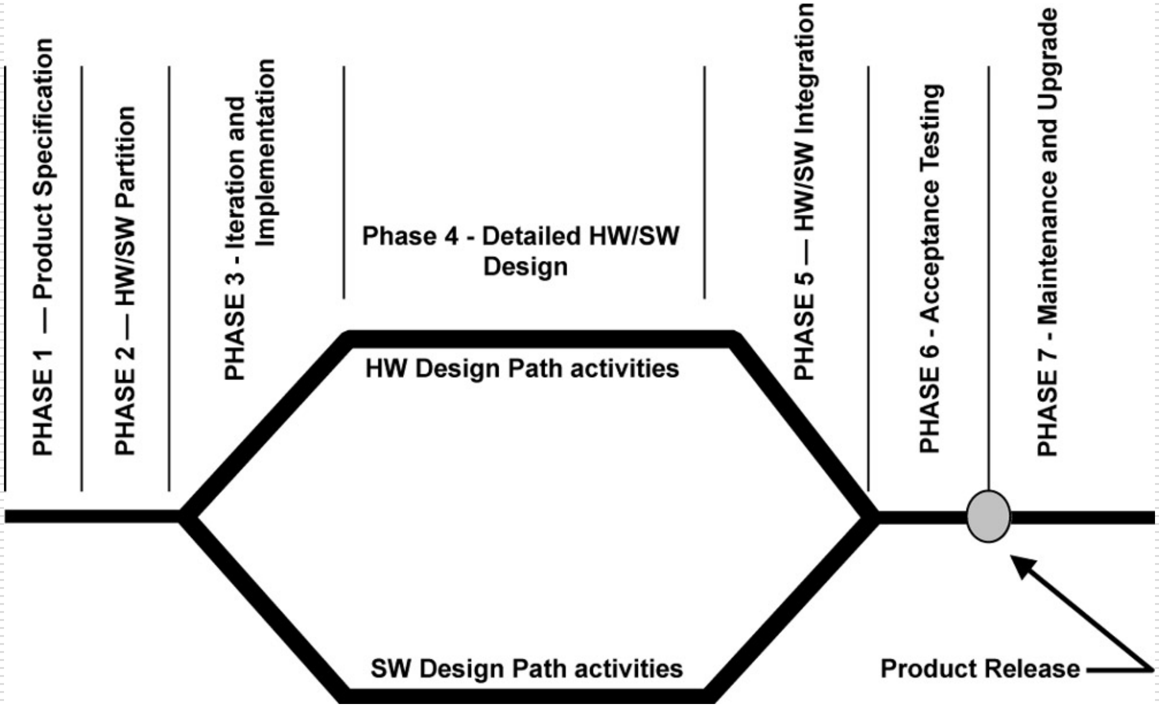
Source: *Embedded Software Strategic Marketing Intelligence Program 2004, Venture Development Corporation.*

Software Development Life Cycle
Realistic Software Life Cycle



Software Development Life Cycle
ERT System Life Cycle

- A schematic representation of the ERT life cycle.


PHASE 1 — Product Specification
PHASE 2 — HW/SW Partition
PHASE 3 - Iteration and Implementation
Phase 4 - Detailed HW/SW Design
HW Design Path activities
SW Design Path activities
PHASE 5 — HW/SW Integration
PHASE 6 - Acceptance Testing
PHASE 7 - Maintenance and Upgrade
Product Release

Software Development Life Cycle
ERT Software Development

- This formalized strategy defined a clear framework for software engineers during the software development process.

- The final quality of ERTS depends upon the development process, the description models, the techniques and tools used (Calvez, Pasquier and Peckol, 1997).

- Due to the complexity and nature of ERT systems, Mutos and Rodd, (1994) suggested a proper framework and life cycle for ERT systems need to be developed.

- The framework should describe, explicitly, all the possible interactions within the final, overall system – including the target real-world application.

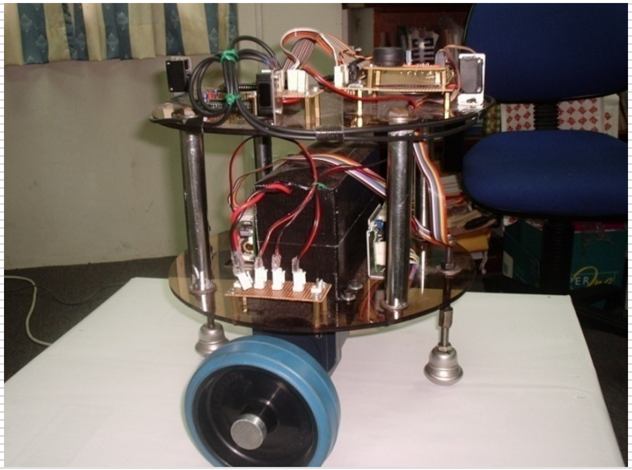
Software Development Life Cycle
Requirements Specification

- Issues that must be addressed in the requirement specification phase for ERT system are :
  1. functional modeling,
  2. behavior modeling,
  3. timing,
  4. man-machine interface and
  5. hardware-software interface.

Software Development Life Cycle
Requirements Specification (an example)

Autonomous Mobile Robot Requirement

The AMR consists of a body and a pair of wheels. Each drive wheel is move by a direct-current (DC) motor. The speeds of the motors are sensed using shaft encoders and fed back to the embedded controller for computation of control signal to the DC motor every 100 milliseconds using the proportional-integral (PI) control algorithm. The embedded controller also monitors the robot environment using four infrared (IR) proximity sensors and communicates with human using Liquid Crystal Display (LCD) and switches.



Software Development Life Cycle
Architecture & Design

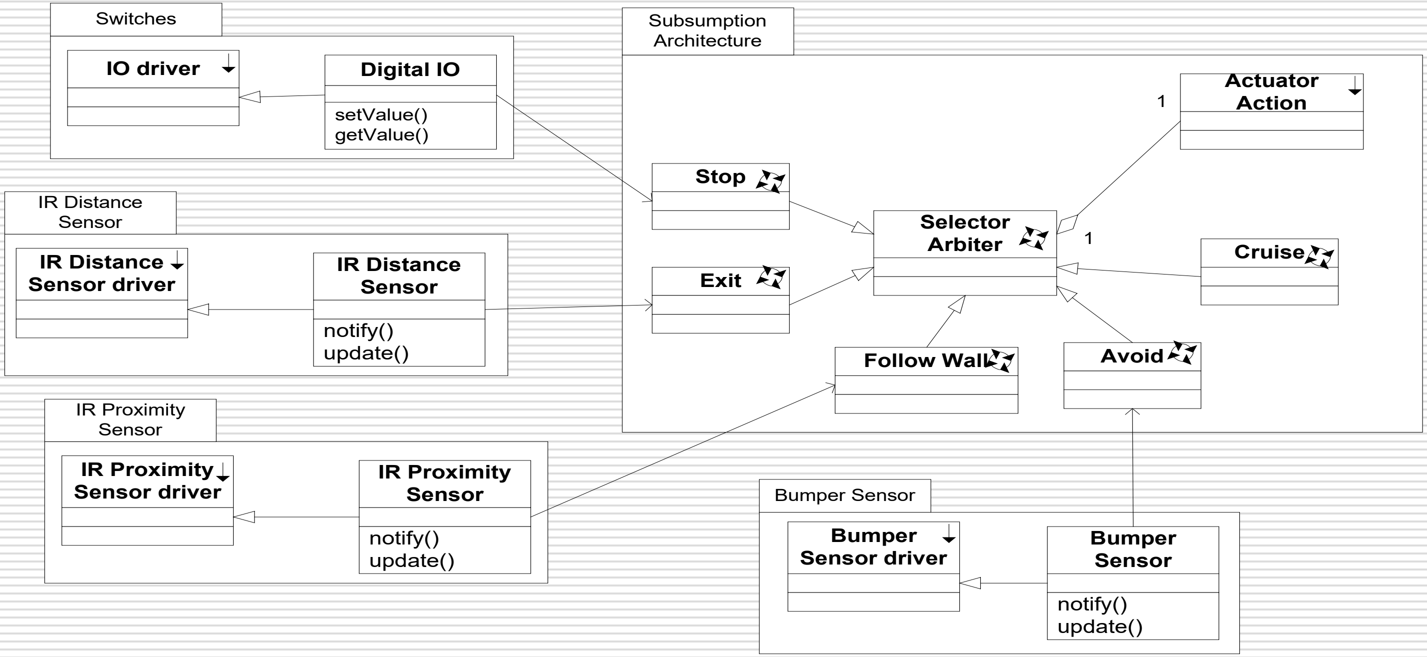
- The architectural design of a system identifies the key strategies for the large-scale organization of system under development.

- These strategies include the mapping of software packages to processes, bus, and protocol selection, and the concurrency model & task threads.

- Issues in ERT design:
  1. Modeling – be able to visualize all possible requirements & predict the RT behavior timeliness at early stage of development.
  2. OO technology - fit well in real-time system problems, this combination is used in some software engineering methodologies such as HRT-HOOD systems.
  3. Scheduling - strategy depends on facilities that the chosen RTOS offers.

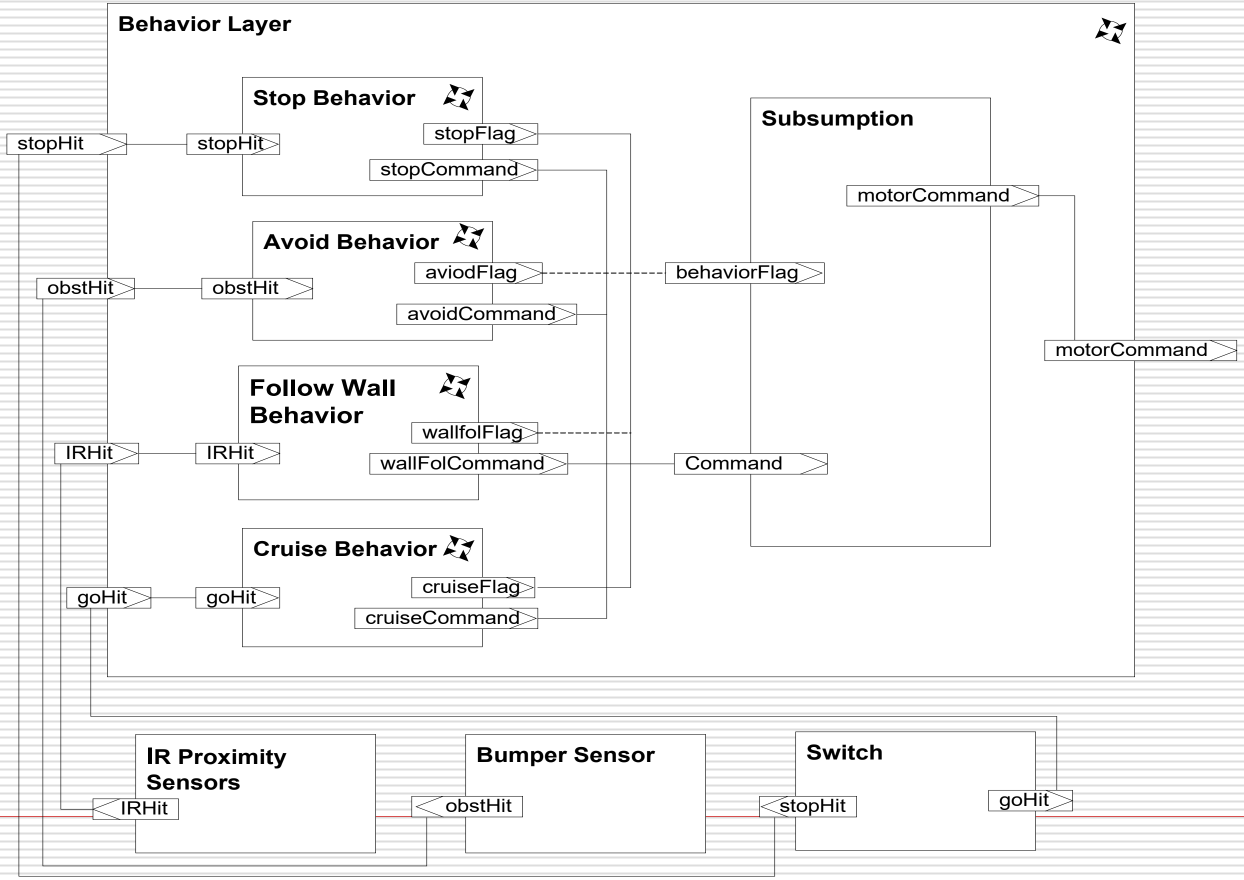
Software Development Life Cycle
Architecture & Design (an example)

- OO technology – A detail internal classes representation of the AMR for five packages.



Software Development Life Cycle
Architecture & Design (an example)

- Architecture structure of the components in AMR design.



Software Development Life Cycle
Architecture & Design (an example)

- Example: the scheduling policy used in the AMR system is Preemptive Priority-based Execution.

Behavior	Wcet (*C*)	Period (*T*)
Avoid$_{exec}$	100	500
Follow$_{exec}$	100	500
Avoid$_{sync}$	10	500
Follow$_{sync}$	10	500
Stop$_{sync}$	10	500
Cruise$_{sync}$	10	500
Subsum$_{exec}$	20	500
Stop$_{exec}$	10	1000
Cruise$_{exec}$	20	1000

Based on the scheduling policy, the timing verification at design stage can be perform using Rate Monotonic Analysis (RMA) theory.

Software Development Life Cycle Architecture & Design (an example)

Example of RMA Theorem

A set of n periodic tasks using an appropriate real-time synchronization protocol will always **meet its deadlines**, for all task phasing, iff $\forall, 1 \le i \le n,$

$$\min_{(k,l) in R_i} \sum_{j=1}^{i} C_j \frac{1}{lT_k} \left\lceil \frac{lT_k}{T_j} \right\rceil \le 1$$

where C_i, T_i, and B_i, are the worst-case execution, period and blocking time for a task *i*, and

$$R_i = \{(k,l) \mid 1 \le k \le i, l = 1, \ldots., \left\lfloor \frac{T_i}{T_k} \right\rfloor \}$$

Software Development Life Cycle
Implementation

- ERT can be implemented on a single or multi processor and multi tasks are designed to run on one single processor.

- To implement multitasking or concurrency in ERTS, 2 software engineering principles need to be considered (Burn and Welling, 1996) are:
  1. real-time design methodology, which leads naturally into multitasking or concurrency implementation.
  2. how to undertake or support multitasking and concurrency in the implementation stage. There are 2 ways to support the factor either
     i. using concurrent real-time language (e.g. ADA) or
     ii. use a sequential language together with a RTOS.

Software Development Life Cycle
Validation, Verification and Testing

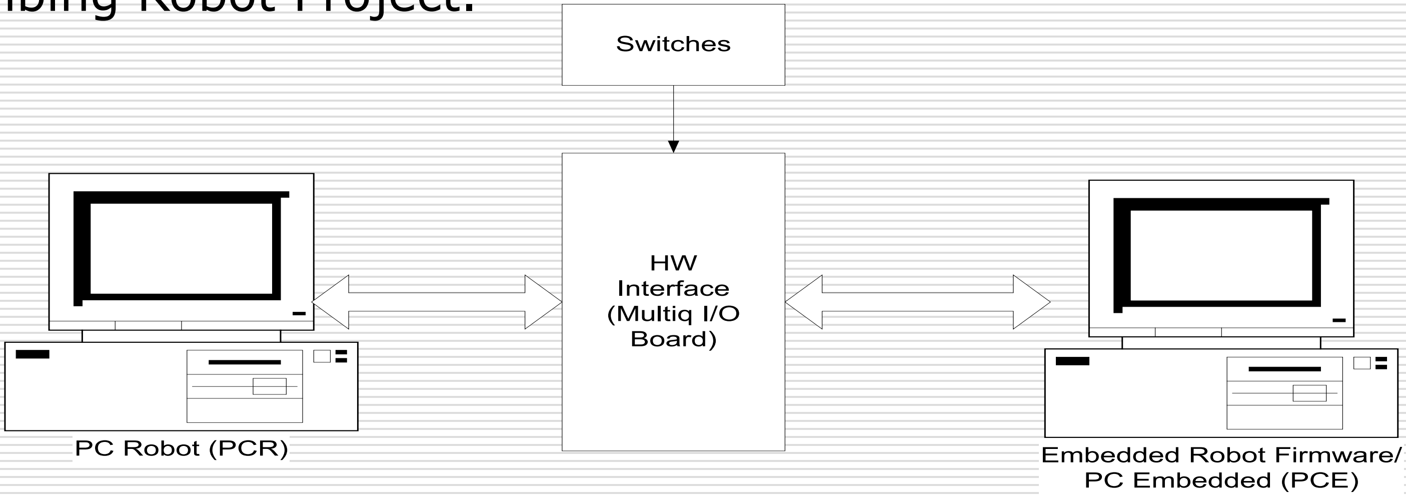
- The ERT software developers frequently must design and write the complex software for hardware that does not exist and this makes the integration and validation testing become more difficult and lengthy.

- In the absence of target hardware, verification of software code is tested using cross development tool, called simulator.

- A simulator is a program (also called "stub"), which imitates the action of the target hardware of the engineering system where the software is embedded.

Software Development Life Cycle
Validation, Verification and Testing (an example)

- ❖ An example of simulation environment used in our Wall-Climbing Robot Project.


Switches
HW Interface (Multiq I/O Board)
PC Robot (PCR)
Embedded Robot Firmware/ PC Embedded (PCE)

- ❖ Main advantages of the proposed simulation strategy are:
  1. Testing can be done concurrently with the dev. of robot hw.
  2. Communication through the hw interface between 2 PCs provides isolation between dev. of simulation.
  3. Code for hw interface between the two PCs is similar and provide realistic timing to the actual robot HW interface.

Software Engineering Essential for ERT
Interest of software engineer in ERT systems

- ERT software is traditionally the domain of practicing engineers, not research scientist. WHY - The software problem in ERT systems, has been viewed by some as too small and simple, which can be handle by assembly language programming, and too limited by hardware cost.

- Who write ERT software - The domain expertise engineers who are classically trained in domain and they have little background in the theory of computation, concurrency, OOD, OS and semantics.

- SE disciplines are essential for ERT software dev., which it is becoming more complex, modular, adaptive and network aware due to the evolving systems requirements.

Software Engineering Essential for ERT

Tasks of Software Engineer in ERT Systems

- To ensure the success of SE in addressing ERT systems problem, Stankovic, 1996, suggested SE will need to undergo a radical shift :

  1. Time, dependability, and other QoS constraints must become first-class concerns, coherently integrated with functionality at all levels from requirements specification through architecture, design, implementation, and execution.
  2. Evolvability must be ensured by separating platform-dependent concerns from application concerns .
  3. Software must be structured into compose-able modules in which interfaces capture not only functionality.
  4. Timing constraints, in particular on individual components, must be dynamically derived and imposed on the basis of end-to-end requirements.

Conclusion

- ERT systems is an enabling technology for many current and future applications that affect public safety, competitiveness, the economy, and life-style.
- Software for ERT needs to be optimized.
- The contribution of software engineer is required to face the challenge, where the methods used for general-purpose software require considerable adaptation for ERT software
- SE disciplines are essential for ERT software development but require considerable adaptation for ERT software.

References

1. Cooling J. (2003). “Software Engineeing for Real-Time Systems”, Addison-Wesley. .
2. David Tennenhouse, “Proactive Computing”, *Communication of the ACM*, Volume 43, no.5, pp 43-50, May 2000.
3. Gupta, R. K. (1998). “Introduction to Embedded Systems.” *Lecture note ICS212.* University of California.
4. Edward A. Lee, “Embedded Software”, *Advances in Computers*, Vol. 56, Academic Press, September 2002.
5. Dejean Milojicic, 2002, “Trends War – Embedded Systems”, *IEEE Concurrency*, Vol.8, Issue 4, Oct-Dec 2000.
6. Arnold S. Berger **“**Embedded Systems Design: An Introduction to Processes, Tools, and Techniques**”**, *CMP Books*, 2002.
7. Calvez, J. P., Pasquier, O. and Peckol, J. (1997). “Software Implementation Techniques for Hw/Sw Embedded Systems.” *Proceedings of the 5th International Workshop on Hw and Sw Codesign (CODES/CASHE’97).* 49-53.
8. Motus, L. and Rodd, M. G. (1994). “Timing Analysis of Real-time Software.” Oxford, UK: *Pergamon*.
9. John A. Stankovic et al. “Strategic Directions in Real-time and Embedded Systems”, *ACM Computing* Survey, Vol. 28, No. 4, December 1996.
10. Klein M.H., Lehoczky J.P., and Rajkumar R., “Rate-Monotonic Analysis for Real-Time Industrial Computing,” *Computer,* Jan. 1994, pp. 24-33.

Thank You.