

Introduction to Logic (1/2)

Moonzoo Kim

CS Division of EECS Dept.
KAIST

moonzoo@cs.kaist.ac.kr

<http://pswlab.kaist.ac.kr/courses/cs402-07>

Why is logic important to software engineers?

- Mathematics v.s. logic
- Difficulty of building correct software
- Logic in a software development framework
- Necessity of rigorous requirement specification
- A programming language as a specification language?

Mathematics v.s. Logic

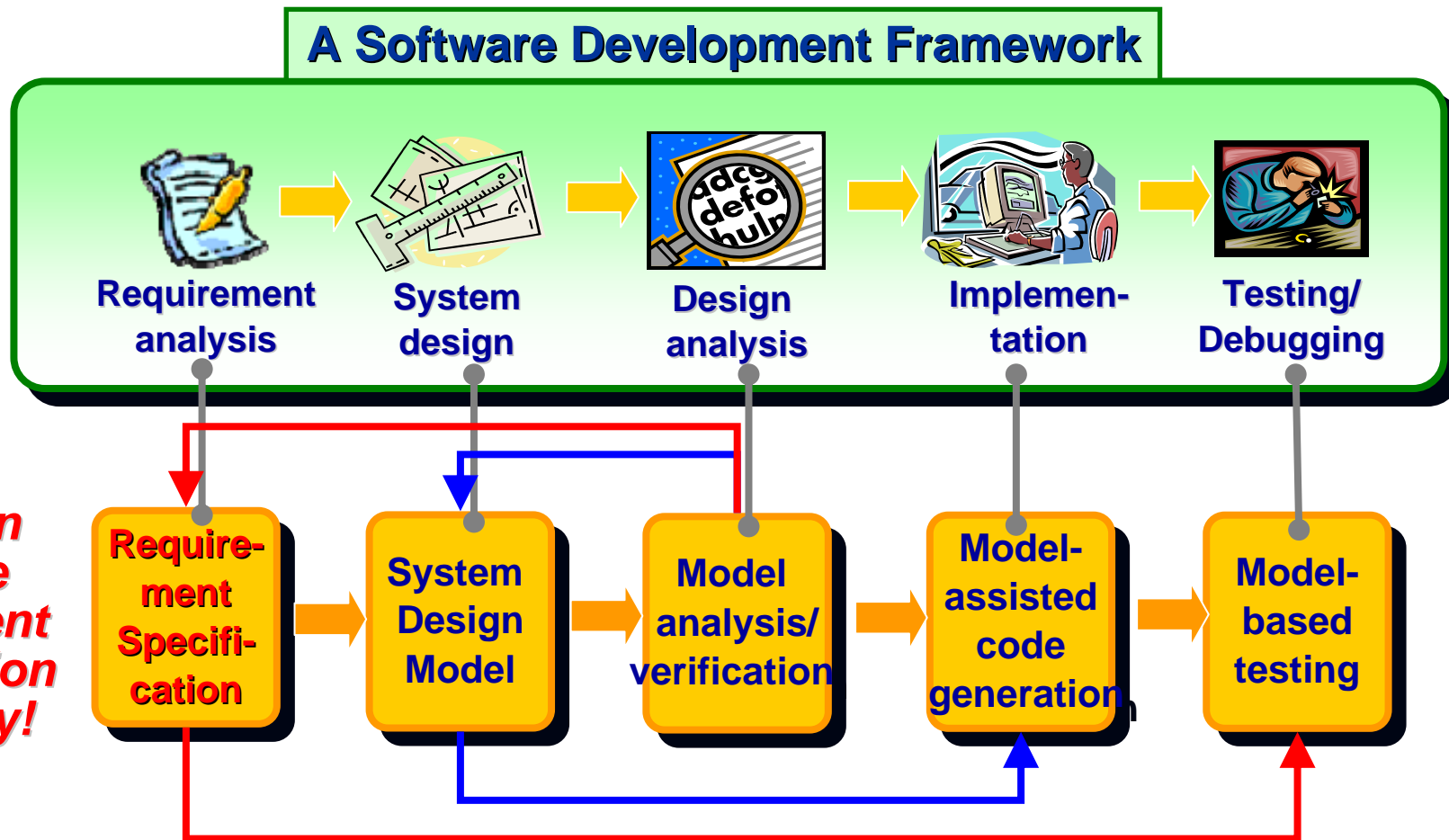
- Mathematics is an essential skill for all engineering fields
 - Electrical engineering, mechanical engineering, chemical engineering, etc
 - The kind of mathematics for the above fields studies **continuous behaviors**
 - Goals are to get a **numeric solution** for equations.
 - Differential equations, integral calculus, probability, etc
 - $$F = \frac{d(mv)}{dt} \quad \int_0^{\infty} \frac{dv}{dt} m^2$$
- Logic is a mathematics for computing systems
 - System engineering, computer science, software engineering, etc
 - Logic studies **discrete behaviors**
 - Goals are to **prove** whether a given logic formula is **true** or **not**
 - Proposition logic, first order logic, temporal logic, etc
 - $p \wedge q, \forall x \exists y x < y$ (*there exists a maximal number*), $\square (x > 10)$ (*invariance*)

(Relative) Difficulty of Building Correct Software

- Most engineering fields can control systems based on feedback because of their continuous behaviors
- Most engineering products handles systems of static structures
- Every engineering field has **well-founded scientific disciplines**
 - Mechanical Engineering
 - Newton's laws
 - Numerical analysis methods
 - Electrical Engineering
 - Circuit algebra
 - Electronic design automation (EDA)
 - Computer Science
 - Mathematical/Computational logic
 - Formal methods
- Computer system **must guarantee their correctness** before deployment, because computer system cannot rely on feedback to avoid failure due to its discrete behavior.
- Software handles systems of dynamic structures
- Products of all engineering fields provide **warranty except computer science**
 - Sonata has 10 years engine warranty
 - Pentium processor has 3 years warranty
 - MS Windows does not provide any warranty
 - Use it at your own risk!!!
- Why?
 - Complexity of SW is far greater than products of other disciplines
 - Failure to follow **scientific disciplines** when developing SW

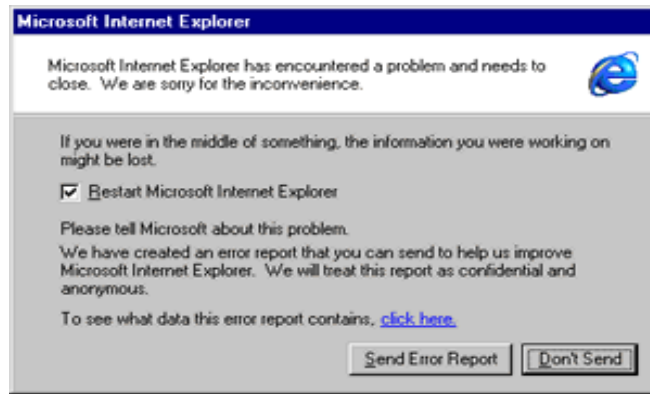
A Software Development Framework

A Software Development Framework



Logic can describe requirement specification rigorously!

Notorious “Blue Screen”



```
*** STOP: 0x0000000A (0x00000000,0x00000002,0x00000000,8038c240)
IRQL_NOT_LESS_OR_EQUAL*** Address 8038c240 has base at 8038c000 - Ntfs.SYS
```

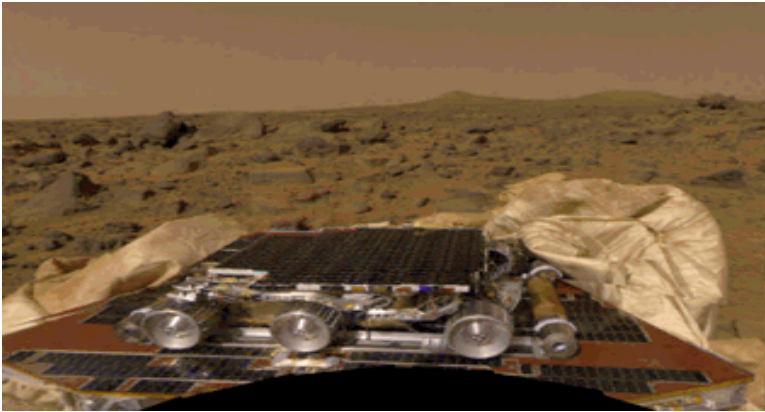
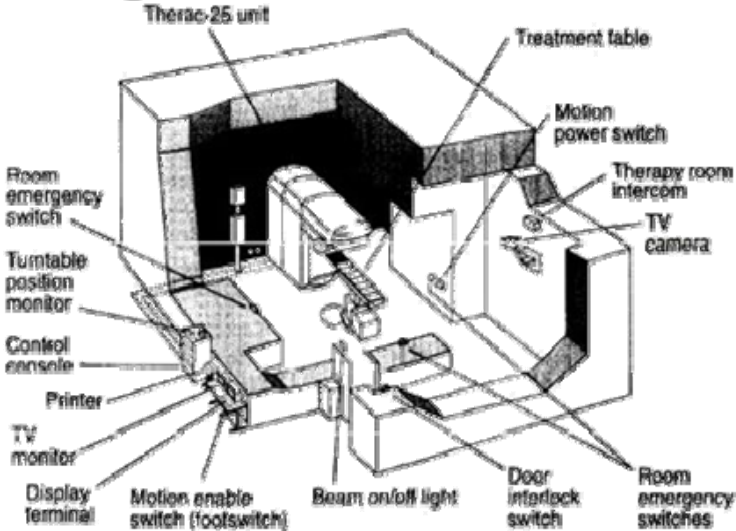
```
CPUID:Genuine Intel 6.3.3 irql:1f SYSVER 0xf0000565
```

Dll Base	DateStmp	- Name	Dll Base	DateStmp	- Name
80100000	336546bf	- ntoskrnl.exe	80010000	33247f88	- hal.dll
80000100	334d3a53	- atapi.sys	80007000	33248043	- SCSIPORT.SYS
802aa000	33013e6b	- epst.mpd	802b5000	336016a2	- Disk.sys
802b9000	336015af	- CLASS2.SYS	8038c000	3356d637	- Ntfs.sys
802bd000	33d844be	- Siwvid.sys	803e4000	33d84553	- NTice.sys
f9318000	31ec6c8d	- Floppy.SYS	f95c9000	31ec6c99	- Null.SYS
f9468000	31ed868b	- KSecDD.SYS	f95ca000	335e60cf	- Beep.SYS
f9358000	335bc82a	- i8042prt.sys	f9474000	3324806f	- mouclass.sys
f947c000	31ec6c94	- kbdclass.sys	f95cb000	3373c39d	- ctrl2cap.SYS
f9370000	33248011	- VIDEOPORT.SYS	fe9d7000	3370e7b9	- ati.sys
f9490000	31ec6c6d	- vga.sys	f93b0000	332480dd	- Msfs.SYS
f90f0000	332480d0	- Npfs.SYS	fe957000	3356da41	- MDIS.SYS
a0000000	335157ac	- win32k.sys	fe914000	334ea144	- ati.dll
fe0c9000	335bd30e	- Fast fat.SYS	fe110000	31ec7c9b	- Parport.SYS
fe108000	31ec6c9b	- Parallel.SYS	f95b4000	31ec6c9d	- ParVdm.SYS
f9050000	332480ab	- Serial.SYS			

Address	dword	dump	Build [1314]	- Name
801afc24	80149905	80149905	ff8e6b8c 80129c2c ff8e6b94	8025c000 - Ntfs.SYS
801afc2c	80129c2c	80129c2c	ff8e6b94 00000000 ff8e6b94	80100000 - ntoskrnl.exe
801afc34	801240f2	80124f02	ff8e6df4 ff8e6f60 ff8e6c58	80100000 - ntoskrnl.exe
801afc54	80124f16	80124f16	ff8e6f60 ff8e6c3c 8015ac7e	80100000 - ntoskrnl.exe
801afc64	8015ac7e	8015ac7e	ff8e6df4 ff8e6f60 ff8e6c58	80100000 - ntoskrnl.exe
801afc70	80129bda	80129bda	00000000 80088000 80106fc0	80100000 - ntoskrnl.exe

Restart and set the recovery options in the system control panel or the /CRASHDEBUG system start option. If this message reappears, contact your system administrator or technical support group.

Tragic Accidents Caused by SW Bug





June 2002

“Software bugs, or errors, are so prevalent and so detrimental that they cost the U.S. economy an estimated \$59.5 billion (60조원) annually, or about 0.6 percent of the gross domestic product

...

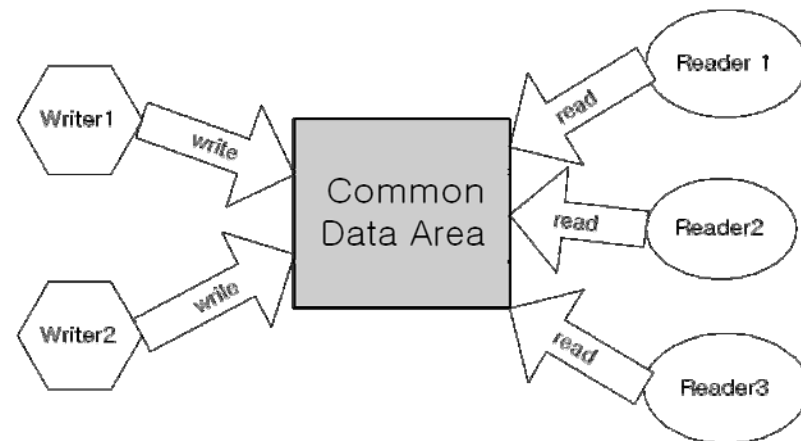
At the national level, over half of the costs are borne by software users and the remainder by software developers/vendors.”

...

The study also found that, although all errors cannot be removed, more than a third of these costs, or an estimated \$22.2 billion, could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects.”

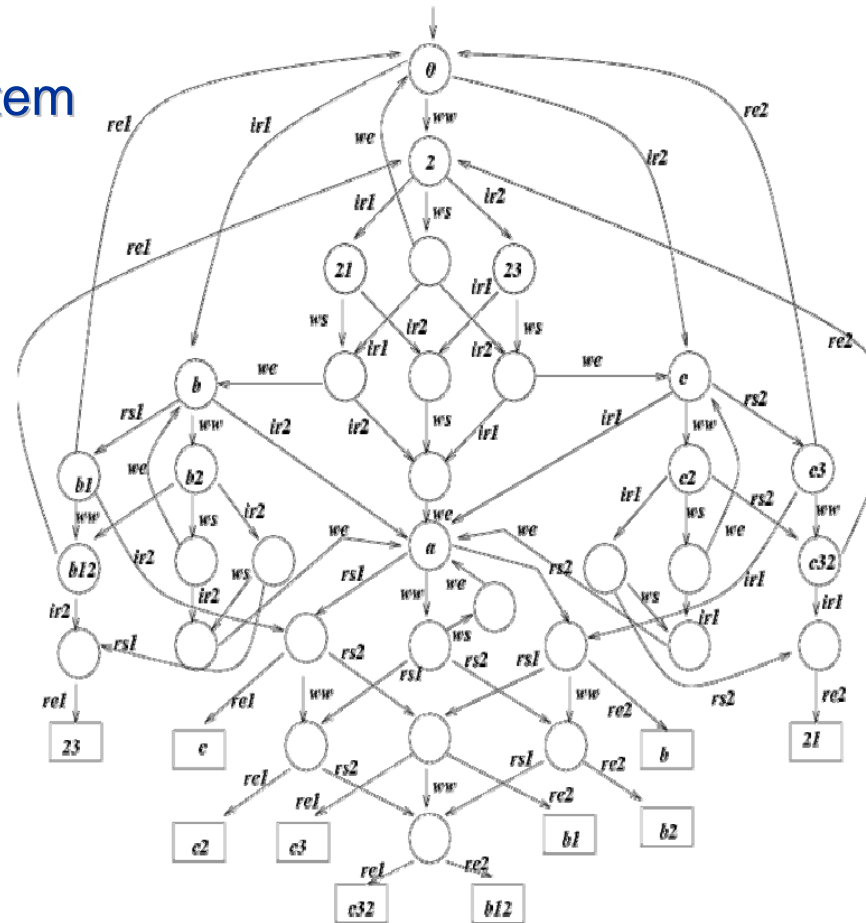
Requirement Specification

- Requirement specifications are the **goals** that a target software must satisfy
- Ex> For a system containing 3 readers, 2 writers, and common common data area, the system should satisfy the following three requirement properties
 - **Concurrency (CON)**
 - Multiple readers can read data concurrently
 - **Exclusive writing (EW)**
 - A writer can write into the data area at an instant with no readers
 - **High priority of a writer (HPW)**
 - A writer's request should have a higher priority than that of a reader



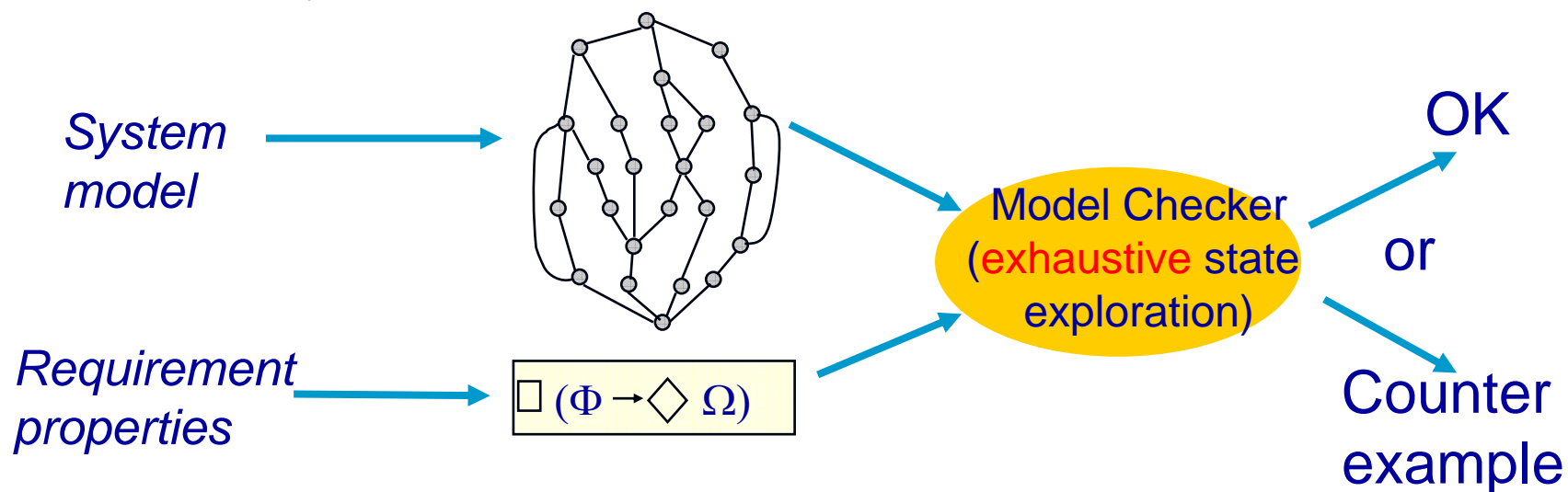
System Design Model

- Abstract description of a target system
- Model must have clear meaning/semantics
- Ex. A system design model for 2 readers and one writer in process algebra
 - System = $(R_1 | R_2 | W | D) \backslash \{ \dots \}$
 - $R_i = \text{check_lock. read_request}_i. \text{start_read}_i. \text{end_read}_i. R_i$
 - $W = \text{check_lock. write_request. start_write. end_write. W}$
 - $D = \text{'read_request}_1. \dots$



Design Analysis

- We definitely want to guarantee/prove that a system design model M satisfies a requirement property ϕ
 - $M \models \phi$



Necessity of Rigorous Requirement Specification 1/4

- Specifications in natural language (ex. 한국어, English, 日本語, etc) can be easily **ambiguous**, **inconsistent**, and **incomplete**
- Connectives in natural languages can have several meaning
 - John eats an apple **and** Mary eats an orange
 - John drove on **and** hit Mary *“and” may have an ordering information in time*
 - The sun is shining **and** John feels happy *“and” may indicate a cause and result*
 - John is a Korean **or** a computer scientist
 - John is working **or** he is at home *“or” may have an exclusive meaning*

Necessity of Rigorous Requirement Specification 2/4

- Specification in natural languages can be easily **incomplete** due to assumption of implicit “common senses” which actually do not exist
- Ex. For the 3-readers and 2-writers system in the page 8
 - Concurrency (CON): “Multiple readers can read data concurrently.”
 - What if only 2 readers can read concurrently?
 - Exclusive writing (EW): “A writer can write into the data area at an instant with no readers”
 - What if two writers write into the data area at the same time?
 - High priority of a writer (HPW): “A writer’s request should have a higher priority than that of a reader”
 - What if a writer requests one second later than a reader? Should the system wait for handling readers request? If so, how long?

Necessity of Rigorous Requirement Specification 3/4

- Informal specification can easily result in erroneous systems
- Example (retail chain management software)
 - If the sales for the current month are below the target sales, then a report is to be printed, unless
 - the difference between target sales and actual sales is less than half of the difference between target sales and actual sales in the previous month
 - or if the difference between target sales and actual sales for the current month is under 5 percent.
 - Find two ambiguous points

Necessity of Rigorous Requirement Specification 4/4

- Suppose that R_i means i th reader is reading.
- The requirement ϕ_{CON} for concurrency can be written in **propositional logic**
 - If it is ok for two readers to read concurrently, ϕ_{CON} should be
 - $(R_1 \wedge R_2) \vee (R_2 \wedge R_3) \vee (R_3 \wedge R_1)$ for **some** time instant t
 - Note that if it is ok for **only** two readers to read concurrently,
 $\phi_{CON} = (R_1 \wedge R_2 \wedge \neg R_3) \vee (\neg R_1 \wedge R_2 \wedge R_3) \vee (R_1 \wedge \neg R_2 \wedge R_3)$
 - If all three readers should be able to read concurrently, ϕ_{CON} should be
 - $(R_1 \wedge R_2 \wedge R_3)$ for some time instant t

A Programming Language as a Specification Language?

- It is not impossible to describe a requirement specification in a conventional programming language such as C/C++ or Java
- But this approach has several disadvantages
 - Conventional programming languages were designed to specify what to do in **a low-level detail**
 - Complex to describe requirement specifications
 - Conventional PL focuses on describing **step-by-step execution (i.e., imperatively)**, not goals of executions.
 - Inconvenient to describe requirement specification
- Logics have been designed to specify/declare req. specifications. So use what is designed for its original purpose!!!
- Logic has **formal syntax** and **formal semantics**
- Ex. For propositional logic
 - Syntax: $\phi = p \mid \neg \phi \mid \phi \wedge \phi \mid (\phi)$
 - Semantics:
 - $[p]_s = \text{true}$ if a proposition p is true; false otherwise
 - $[\phi_1 \wedge \phi_2]_s = \text{true}$ if $[\phi_1]_s = [\phi_2]_s = \text{true}$; false otherwise
 - $[\neg \phi]_s = \text{true}$ if $[\phi]_s = \text{false}$; false otherwise