SAFETY ANALYSIS*

OR
SOME THINGS WE HAVE LEARNED FROM A COUPLE OF CASE STUDIES

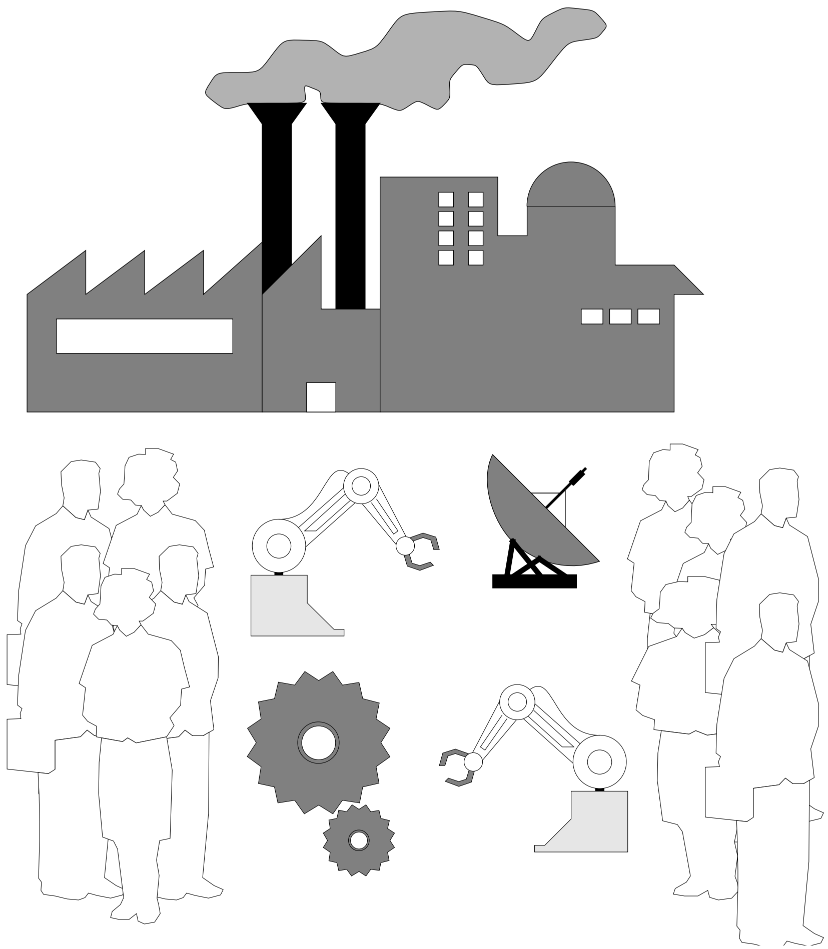
John C. Knight

Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903

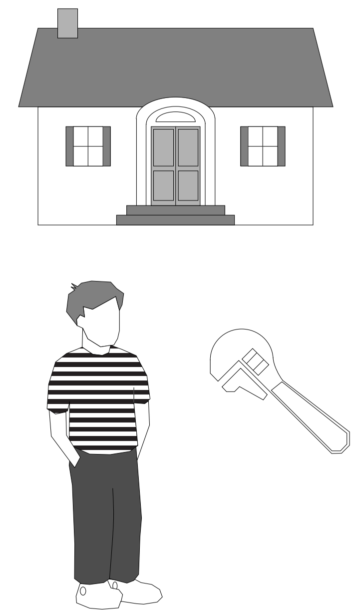
* Work sponsored in part by NASA, the NSF, the NRC, & Motorola Inc

EXPERIENCE WITH SAFETY-CRITICAL SYSTEMS

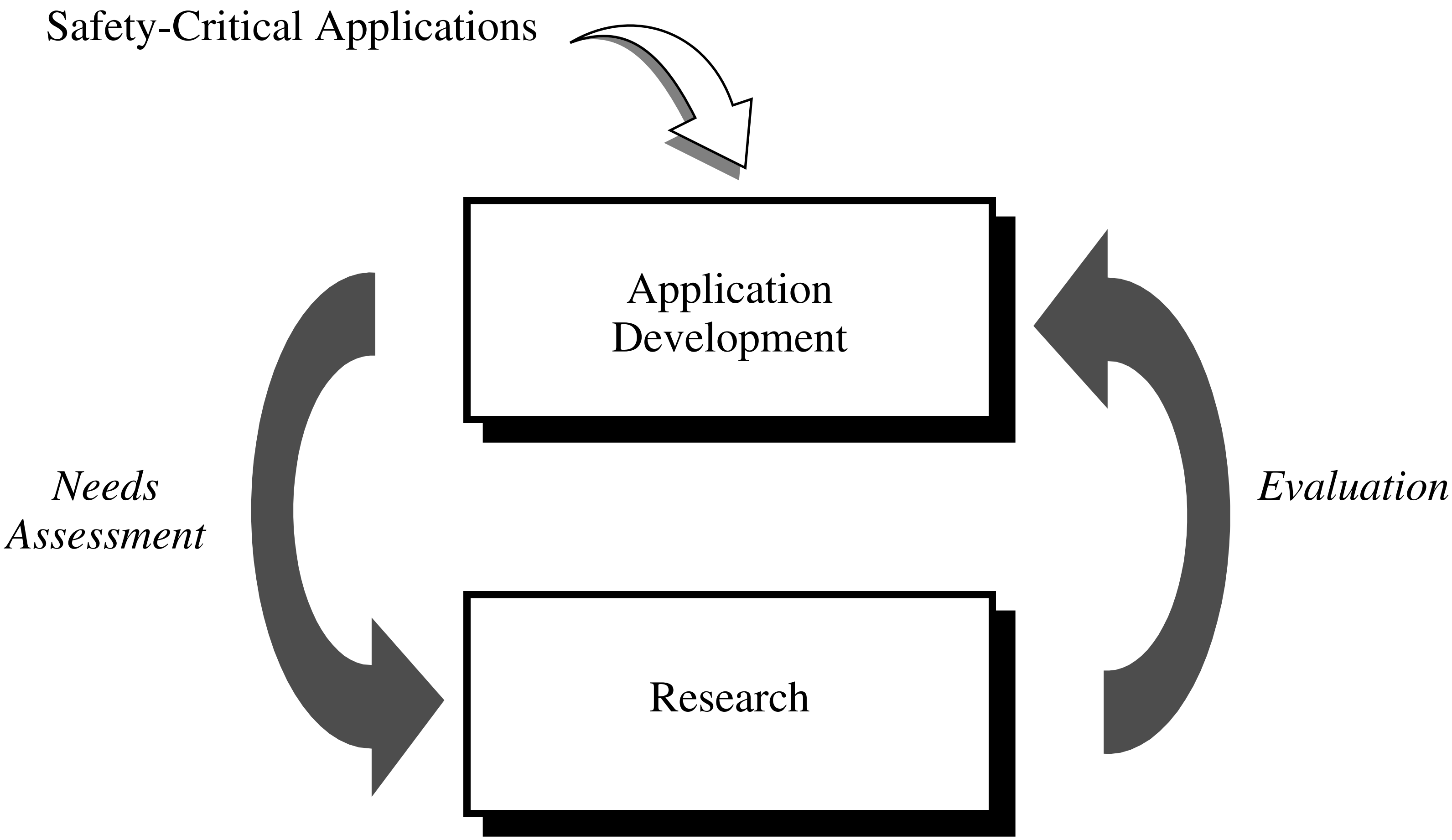
Industry/Government



University

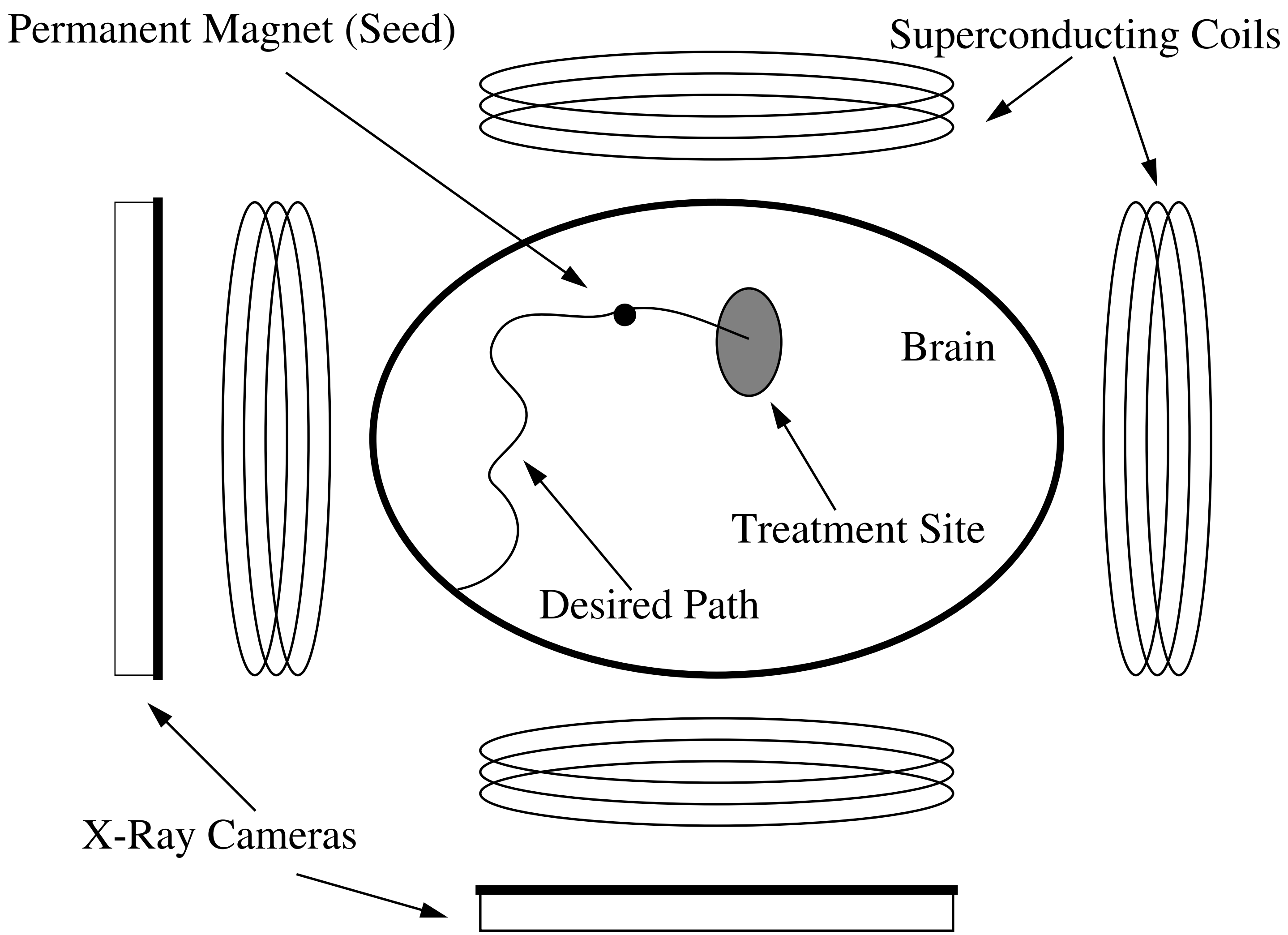


RESEARCH BASED ON CASE STUDIES



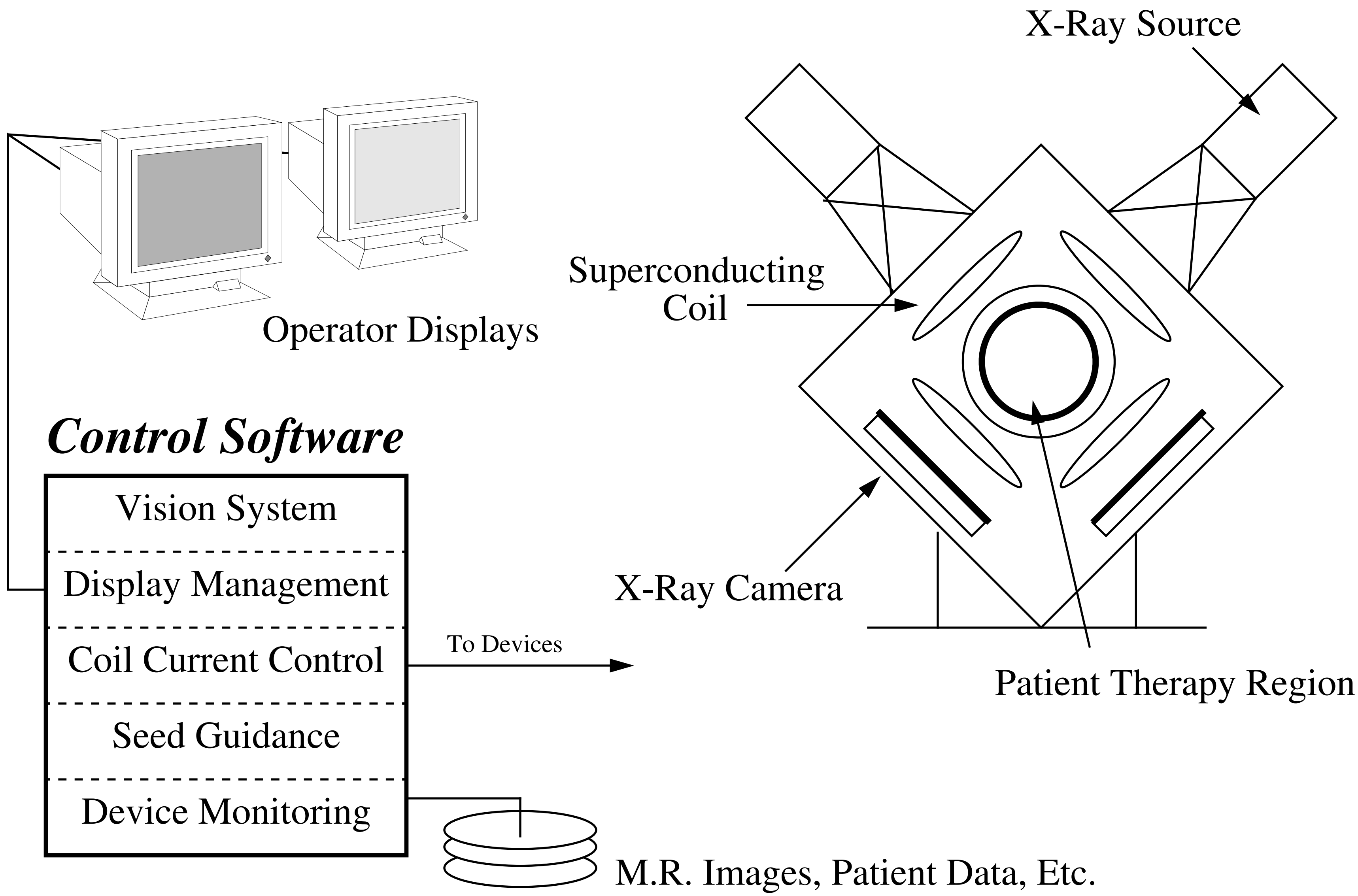
UVA
Department of Computer Science

CASE STUDY MAGNETIC STEREOTAXIS SYSTEM - CONCEPT



UVA
Department of Computer Science

CASE STUDY - MAGNETIC STEREOTAXIS SYSTEM

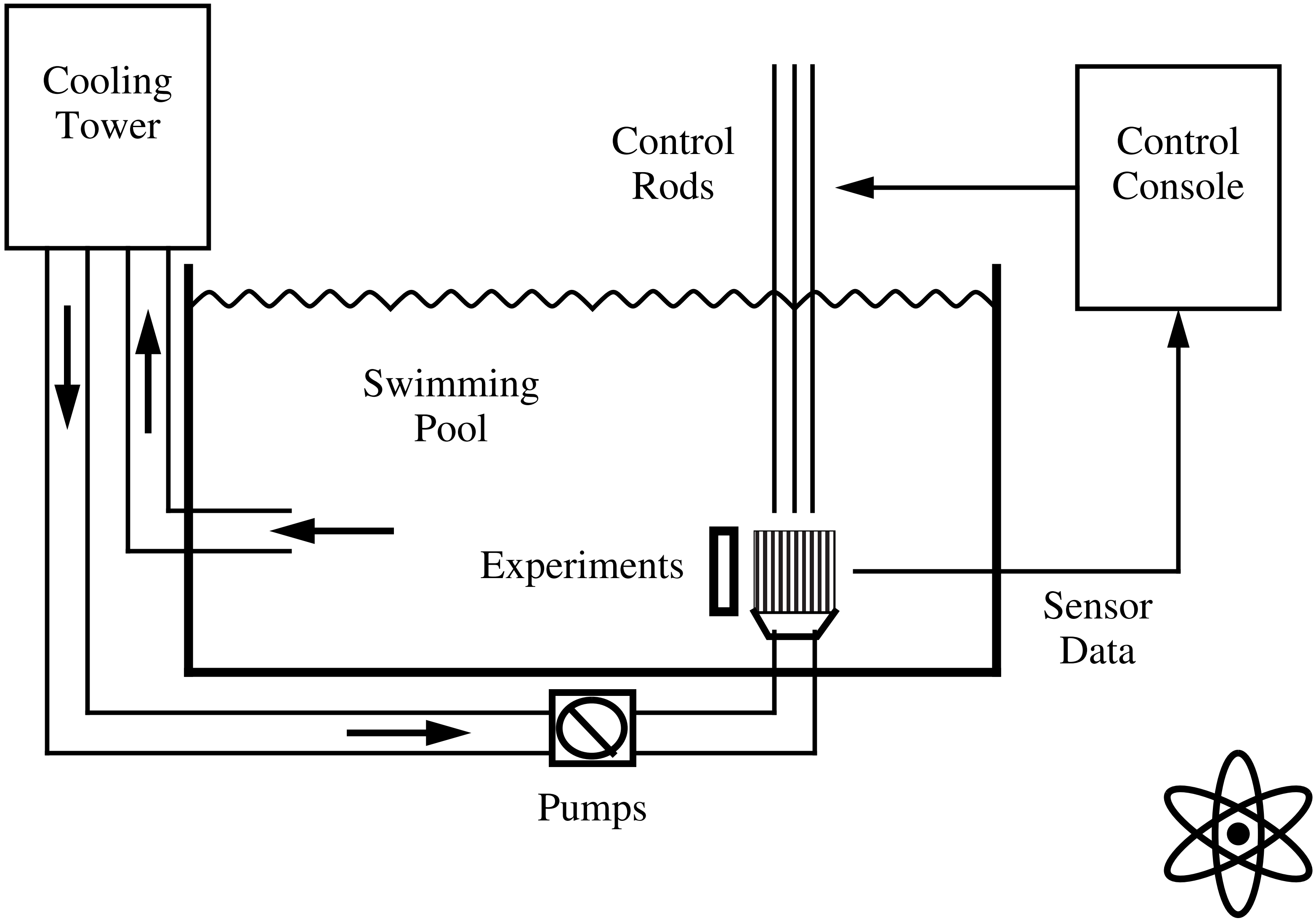


UVA
Department of Computer Science

Some Of The MSS Safety Issues

- External Superconducting Coils:
  - One Or More Coils Fail And Distorted Force Applied To Seed
  - Signals Scrambled Between Computer And Coil Controller(s)
- X-Ray Source Fails “On” When Supposed To Be “Off” Or Vice Versa
- X-Ray Camera Delivers Ghost On Image Or Incorrect Image
- Display System:
  - Wrong Seed Location Shown On Magnetic-Resonance Image
  - Wrong Magnetic-Resonance Image Displayed
- Software System:
  - Erroneous Calibration
  - Incorrect Coil Currents Calculated And Applied
  - Software Commands Any Device “On” Or “Off” Incorrectly
  - Vision System Errors - Skull Marker Or Shadow Observed Rather Than Seed

CASE STUDY - UVA RESEARCH NUCLEAR REACTOR



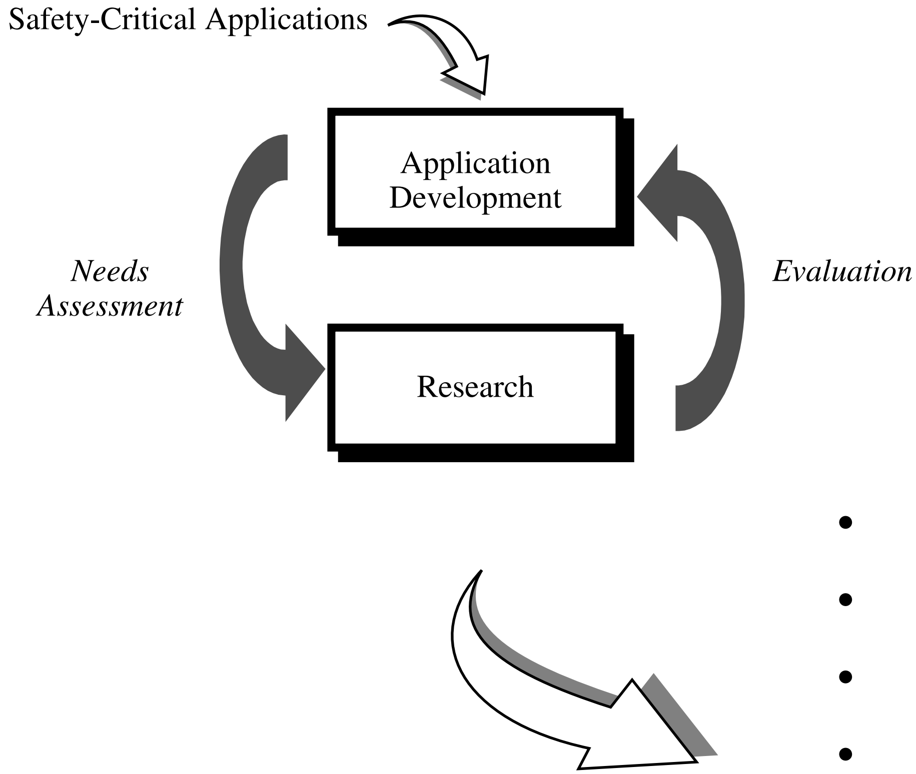
UVA
Department of Computer Science

Some Of The Reactor Safety Issues

- Uncontrolled Withdrawal Of Reactor Control Rods
- Loss Of Water In The Reactor Pool
- Coolant Pump Failure
- High Radiation Levels Outside Of The Reactor Pool
- Undocumented Alteration Of The Core Configuration
- Neutron Flux Sensor Failure
- Air Bubbles Or Other Obstructions To Cooling Inside The Core
- Incompatibility Of Instrumentation And Power Range
- High Power Operation Without Primary Cooling System

UVA
Department of Computer Science

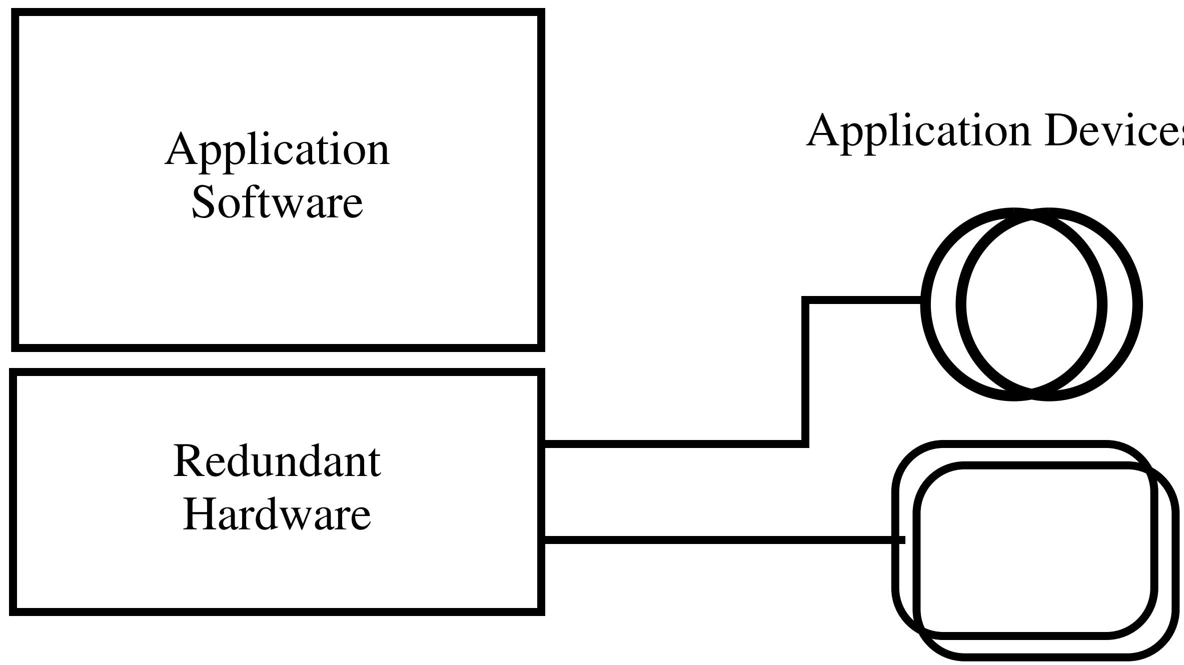
RESEARCH TOPICS



- Safety Specification Capture
- User-interface Specification
- Safety Policy Enforcement
- Software Testing
- Software Reuse
- Software Documentation
- Etc.....

UVA
Department of Computer Science

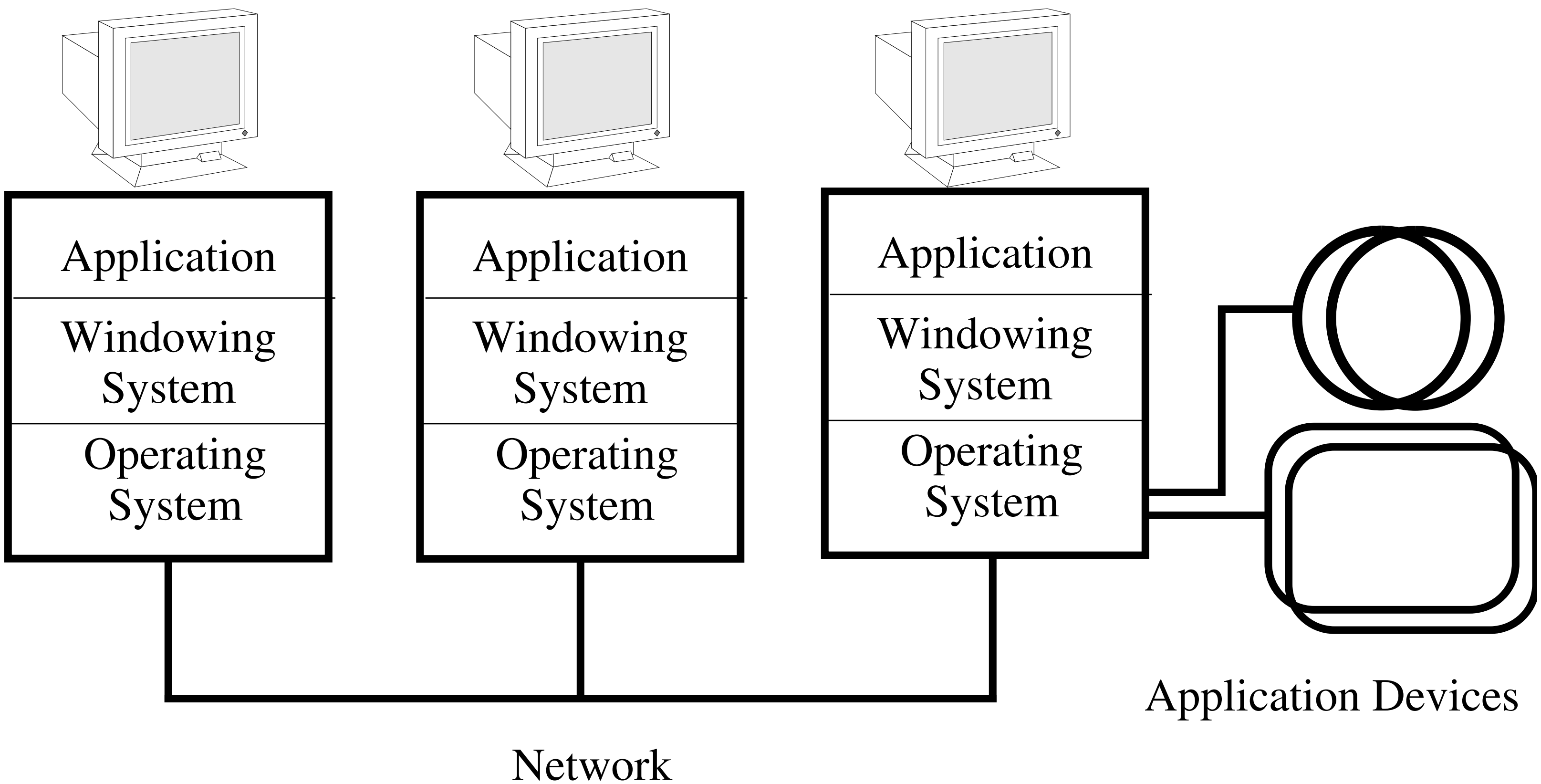
NAIVE APPLICATION ARCHITECTURE



- Keep It Simple, S*****

UVA
Department of Computer Science

REALISTIC APPLICATION ARCHITECTURE

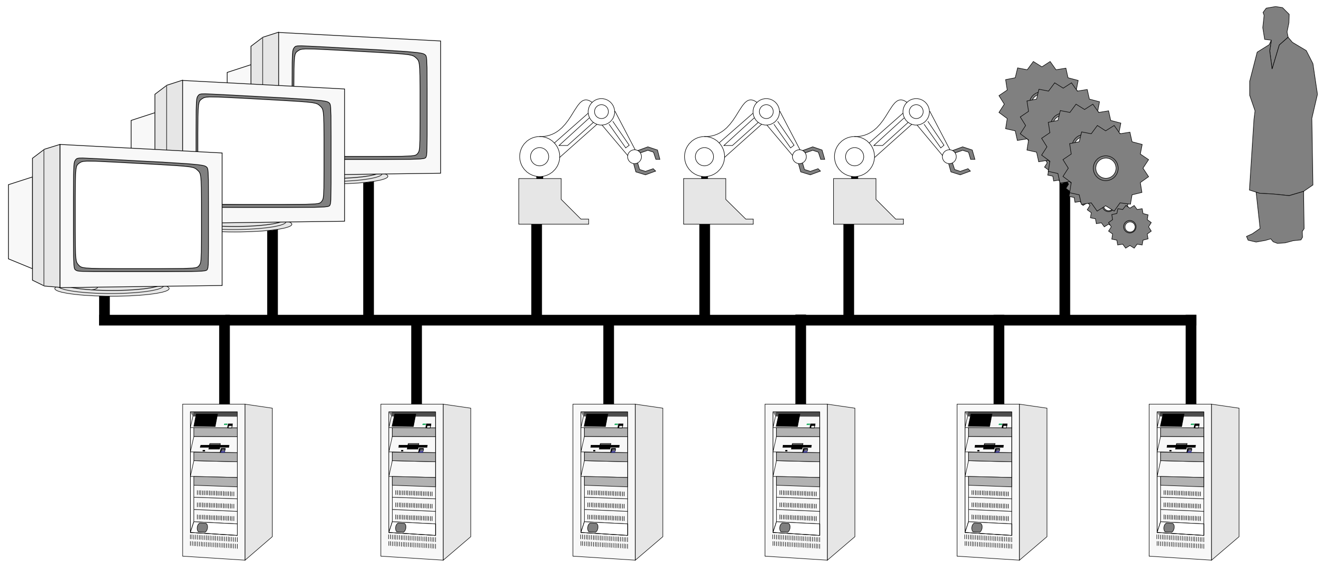


- Diverse Hardware, Network, High-performance Displays
- Extensive, Diverse And Unreliable Software, Perhaps Off-the-shelf

UVA
Department of Computer Science

THE PROBLEM WE FACE

- Software Is *Large And Complex* In Many Safety-critical Systems:



- Huge Subsystems, E.g. System Services, Windowing, The Application, Etc.
- How Do We Build Safety-critical Software That Is:
  - Dependable?
  - Cost-effective?

UVA

Department of Computer Science

SPECIFICATION CAPTURE

Clearly, To Build A System, We Need Some Form Of Formal Specification.

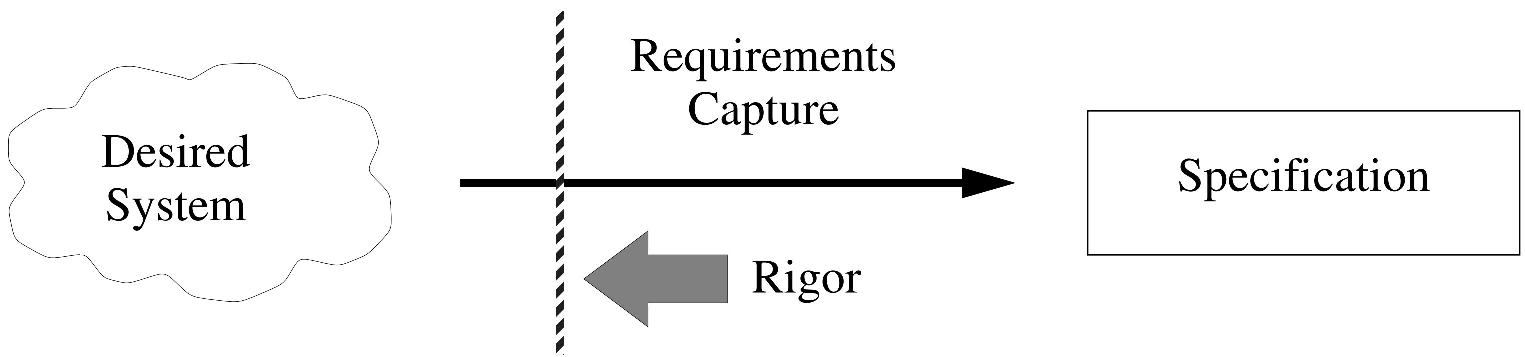
We Chose Z   So We Are Done   Right?

- Approach We (System & Software Engr's) Followed Originally With The MSS:
  - Build The System Fault Trees
  - Develop Formal Safety Specification (In Z)
- The Result Was:
  - An Informal, Ad Hoc Capture Process
  - Continual Revisions To Both The Fault Trees And The Specification
  - Very Little Confidence In The Specification
  - Confusion Over Introducing Software Concerns Into The Fault Trees
  - Confusion Between Software Coping With A Failure And Software Failing
  - Specification That Was Hard To Work With
  - Unsatisfactory Situation...

UVA
Department of Computer Science

REQUIREMENTS CAPTURE - PROCESS GOALS

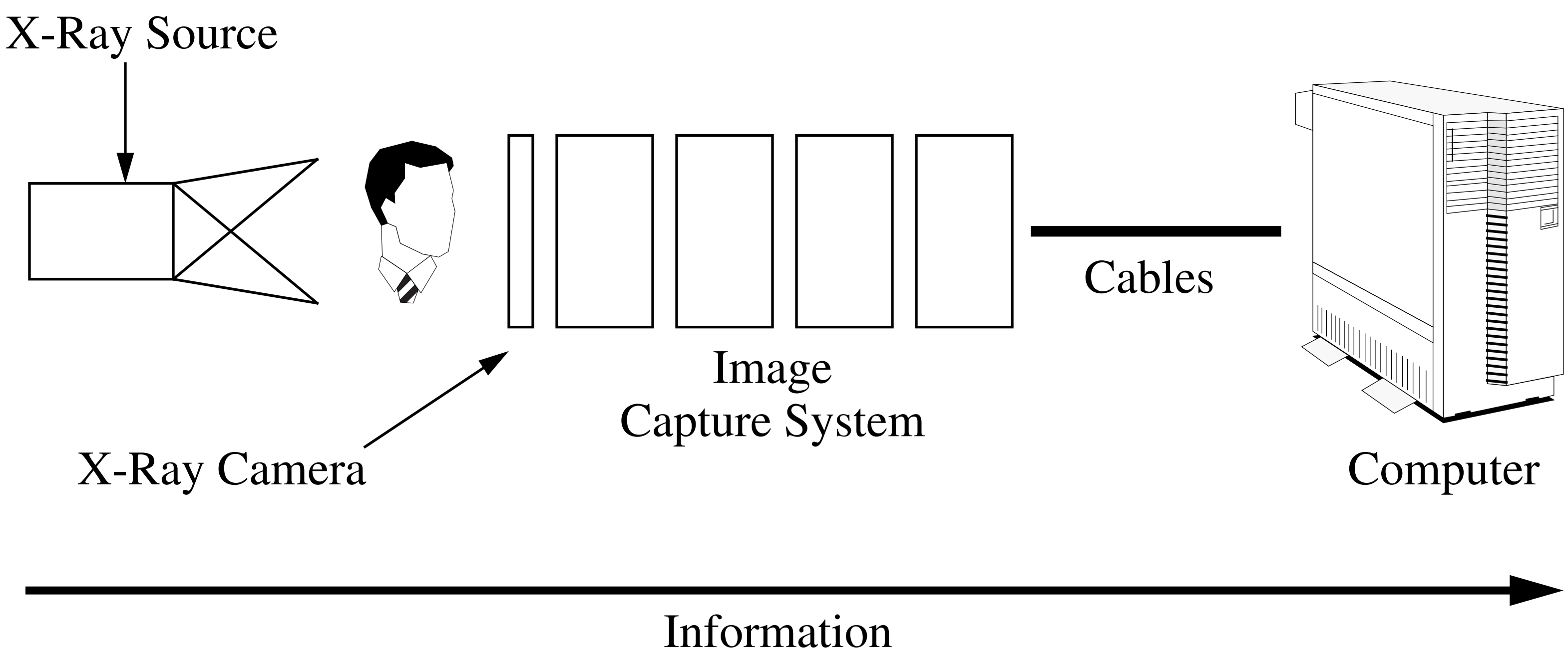
- Separation Of Software Safety And Specification Safety
- Separation Of Software Concerns And Systems Engineering Concerns
- A Process That:
    - Is Rigorous, Repeatable, And Dependable
    - Addresses Real-World Problems
    - Is Accessible To Practitioners
    - Applies To A Wide Spectrum Of Systems
    - Applies At Many Levels Of Formality



UVA
Department of Computer Science

INFORMATION-FLOW MODELLING

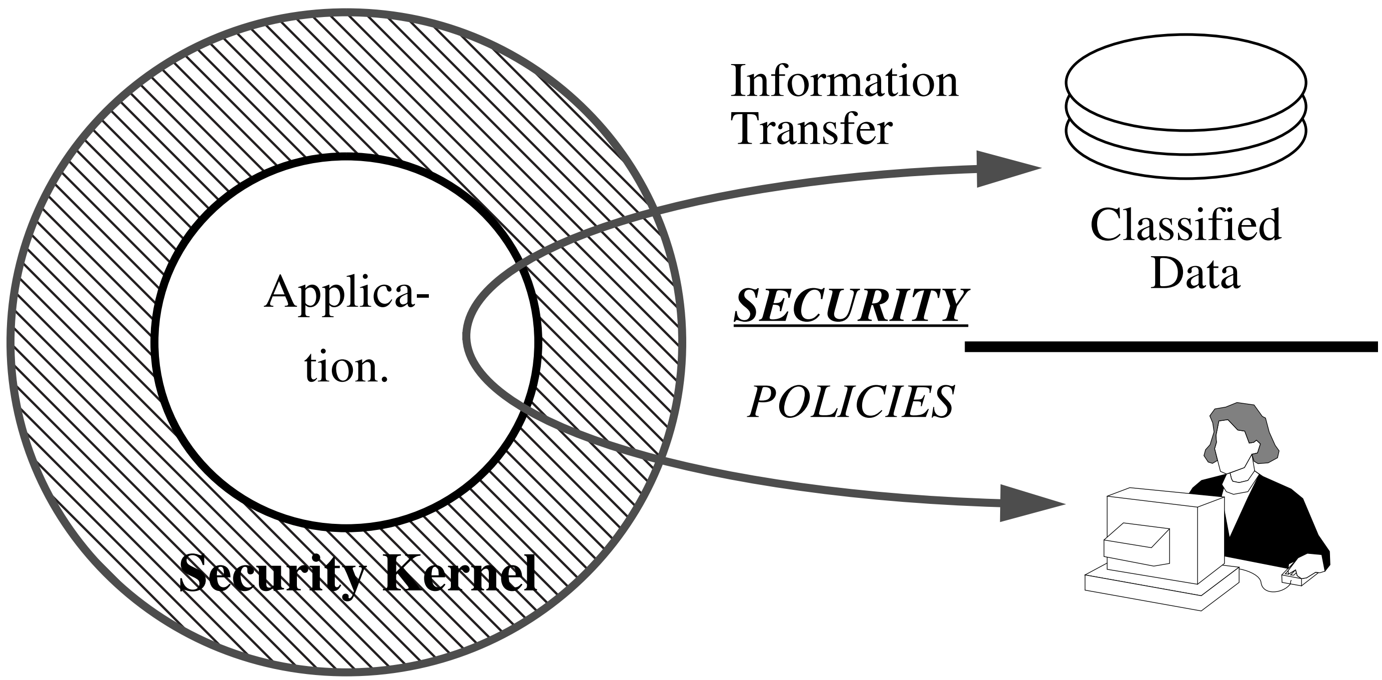
- Models Flow Of Information Through "Information Subsystem"
- Systematically Considers:
  - All Points Where Component (Including Software) Failure Can Corrupt Info.
  - All Points Where Such Failure Can Be Detected & Dealt With



UVA
Department of Computer Science

SECURITY KERNEL CONCEPT

- Concept Is That Security Kernel Controls Access To All Information
- Kernel Enforces A Set Of *Security* Policies Irrespective Of Application Software's Actions:

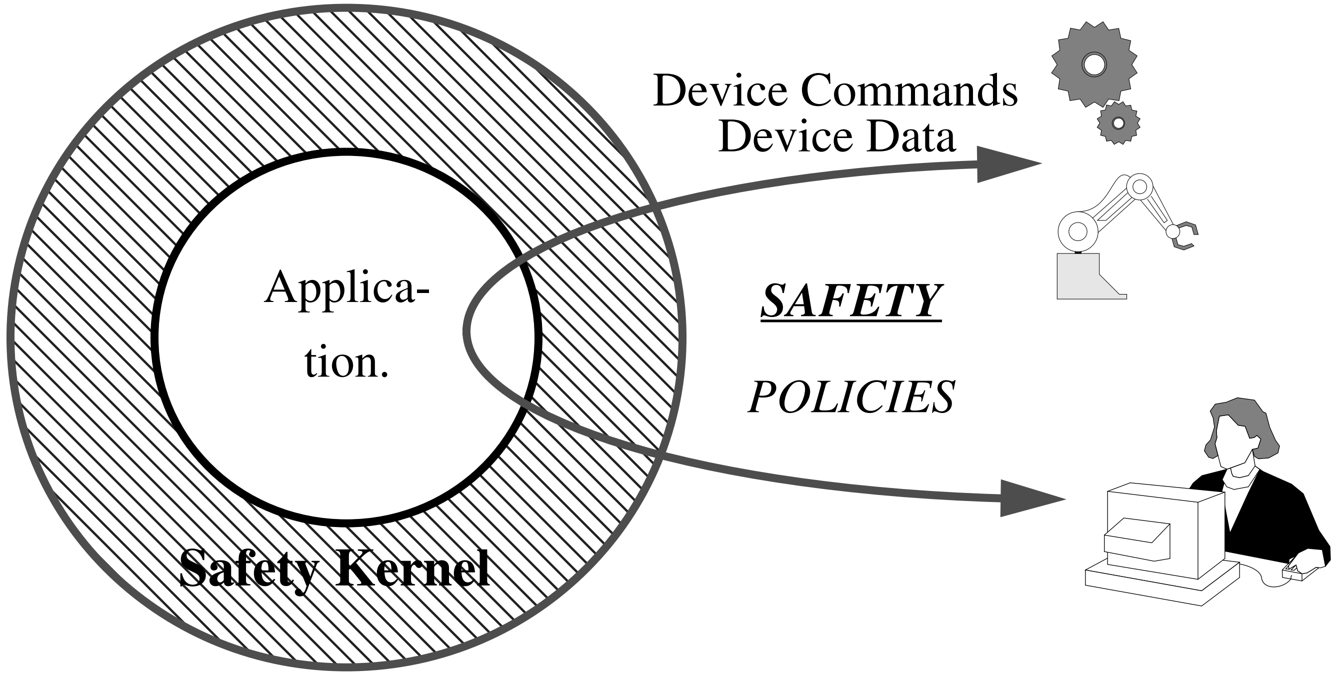


- Might A Similar Approach Work For Safety (Rushby, 1989)?

UVA
Department of Computer Science

SAFETY **KERNEL CONCEPT**

- Concept Is That Safety Kernel Controls Access To All Devices
- Kernel Enforces Set Of *Safety* Policies Irrespective Of Application Software's Actions:



- A Similar Approach Appears To Work For Safety

UVA
Department of Computer Science

Some Of The MSS Safety Policies

If the seed moves faster than 2.0 mm/sec., the coil currents must be set to zero.

The coil currents must be within 5.0 amps of the predicted value.

The coil current requested by the application must be within the range -100 amps to 100 amps.

An X-ray source must be "off" for 0.2 seconds before an "on" command is executed.

The total X-ray dose during an operation must be less than 100 millirem.

Before moving the seed, a reversal check must be executed on the requested currents to compare the predicted force with the desired force.

And so on.....

UVA

Department of Computer Science

Some Of The Reactor Safety Policies

The control rods must not be withdrawn at a rate faster than 1.5 mm/sec.

The position of the regulating control rod must be adjusted at least once per second based on the power of the reactor.

The control rods must be scrammed if a safety channel indicates a power level greater than 125% of the authorized maximum.

The control rods must be scrammed if the pool water level falls below 19’ 3.25”.

The control rods must be scrammed if the inlet water temperature exceeds 105^o F

And so on.....

UVA

Department of Computer Science

SUPPORT VS. ENFORCEMENT KERNELS

SUPPORT KERNEL:

- Support provided for policy enforcement
- Application software responsible for policy enforcement
- Enforcement depends on correct use (or use at all) of kernel operations
- Support provided as potentially useful services, e.g., assertions, timers, device drivers
- Examples: traditional operating systems, some earlier safety kernels

ENFORCEMENT KERNEL:

- Enforces selected policies
- Application software is not responsible for kernel-enforced policies
- Policies enforced irrespective of use of kernel operations
- Kernel not traditional - not necessarily implemented on bare hardware
- Kernel enforces policies much like a fuse enforces a policy in an electrical system

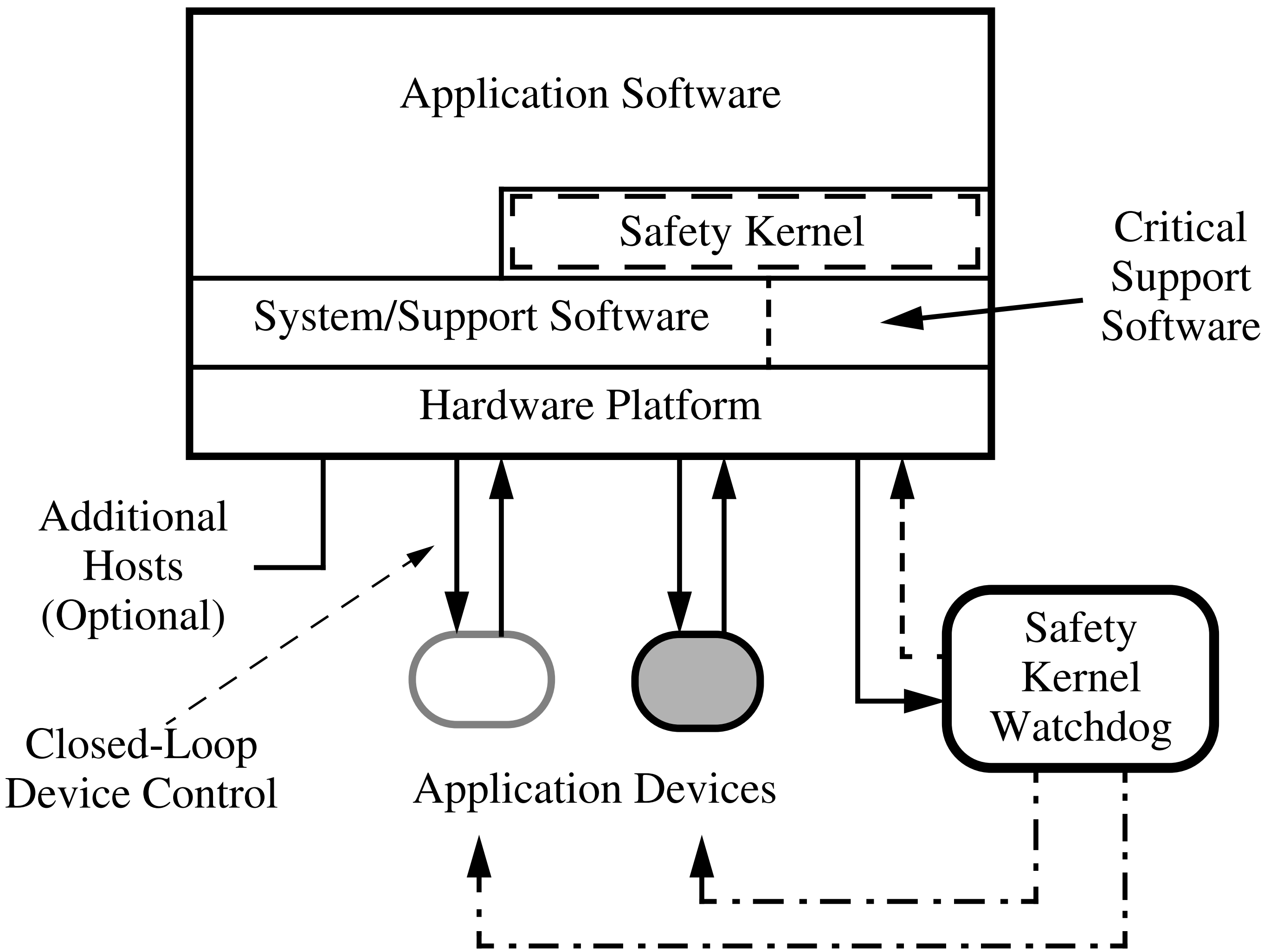
UVA
Department of Computer Science

MAJOR SAFETY KERNEL ISSUES

- How Are Policies Identified?
  - Taxonomy - Rigid Structure
  - Analysis Of Safety Specification
- What Can The Kernel Enforce?
  - Most But Not All Policies
  - Effectiveness Tradeoff
  - “Weakened” Policies
- What Is Required For Dependable Enforcement?
  - Exclusive Device Access
  - Support Services
  - Data Integrity
- How Should Kernel Be Implemented?
- How Can Kernel Be Verified?

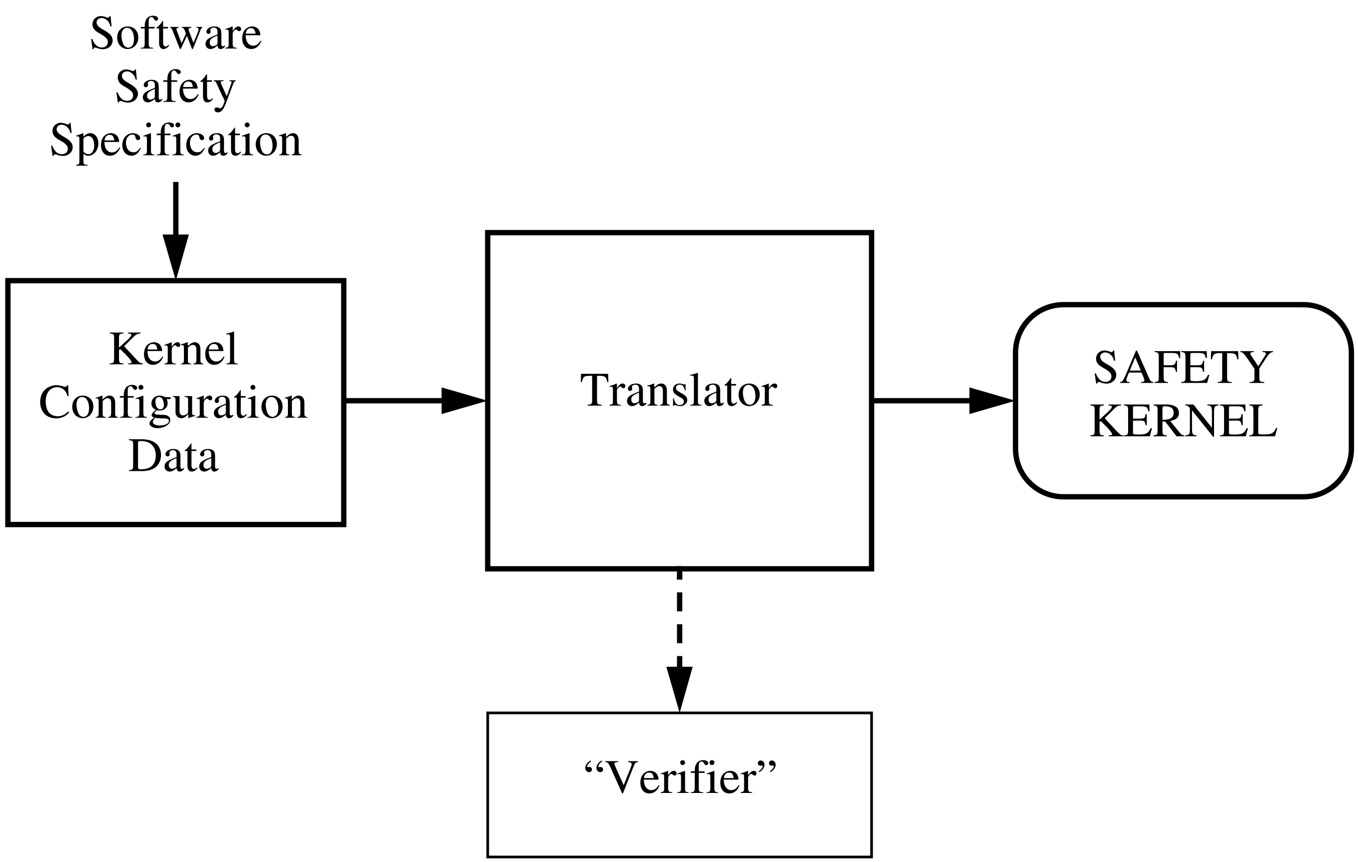
UVA
Department of Computer Science

SYSTEM DESIGN



UVA
Department of Computer Science

SAFETY-KERNEL IMPLEMENTATION



UVA
Department of Computer Science

TESTING ISSUES

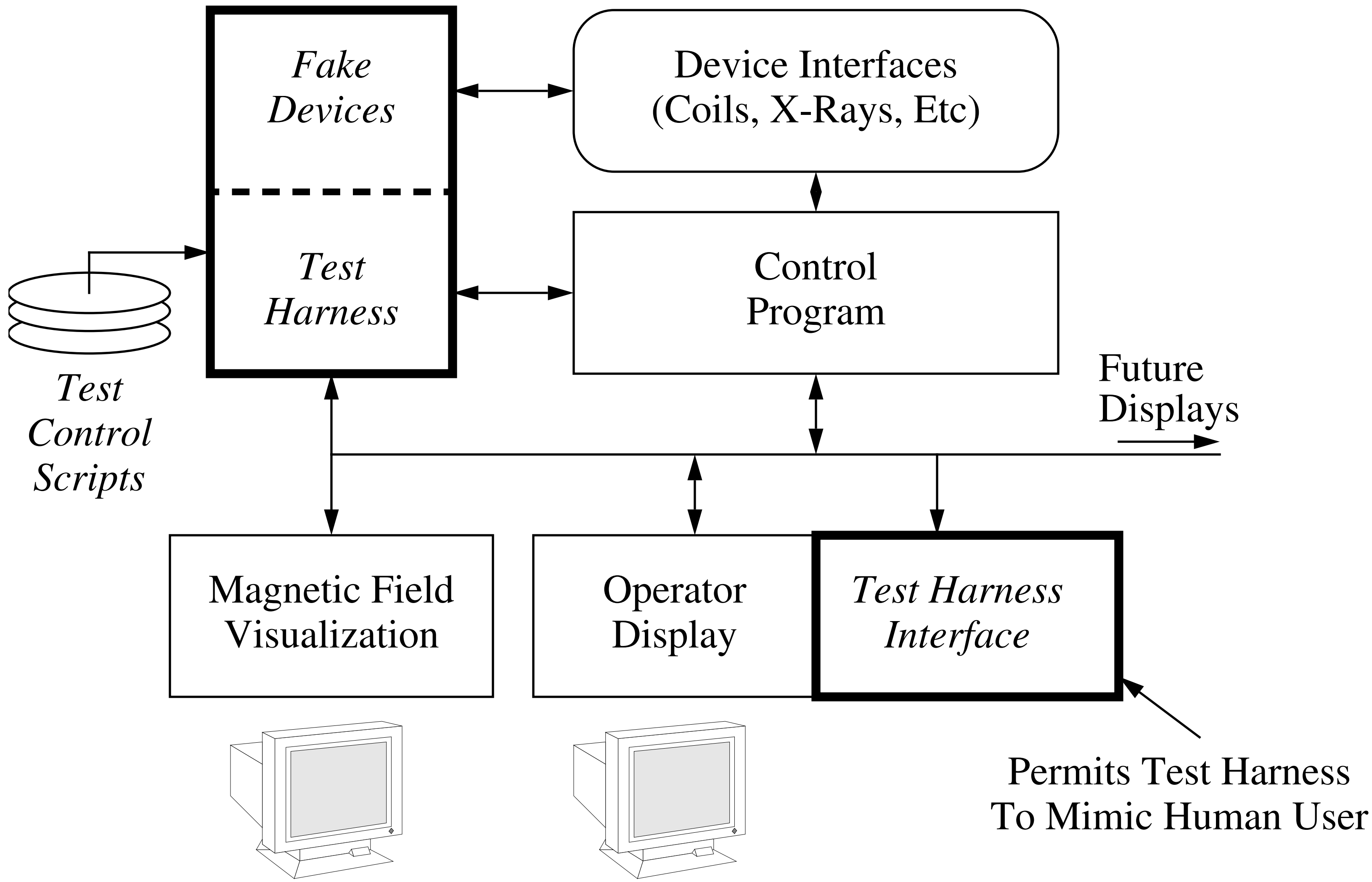
- These Systems:
  - Have Significant Graphic User Interfaces
  - Involve Complex Mathematics
  - Have Stringent Safety Requirements
- Important Research Questions:
  - How Can They Be Tested? They Are Very Complex Systems
  - What Would Testing Mean? We Would Like Meaningful Results
- Testing Safety-critical Systems:
  - Infeasible (Butler & Finelli)
  - *Worse* (Ammann, Brilliant, and Knight)

☞ ***BUT***:

Maybe Restricted But Useful Properties Can Be Formally Established

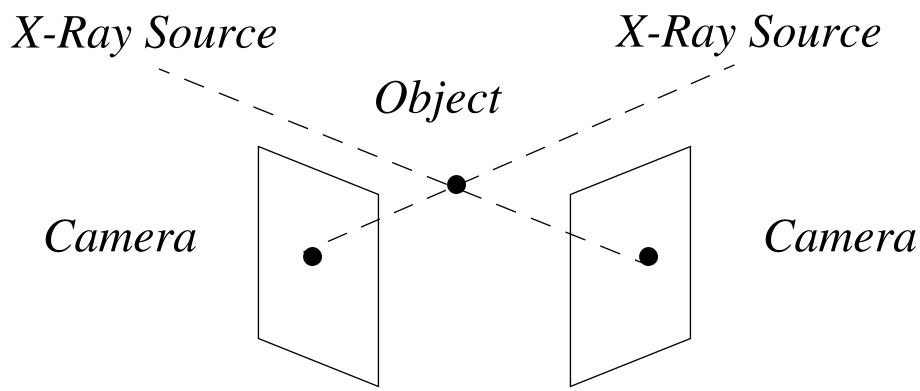
UVA
Department of Computer Science

MSS TESTING ARCHITECTURE



UVA
Department of Computer Science

MSS Test Case Selection



- Vision System Problem:

  Two 2D X-Ray Images, Find Object In 3D Space

- Digitized Images:

  Number Of Object Images ~10^9(Ignoring Background)

- Vision System Can Be Tested With All Possible Object Shadows:
  - Time Required < 1500 CPU Hours
  - A Useful Property Although It's Not "Correctness"

UVA
Department of Computer Science

SPECIFICATION LIMITATION

☞ Deliberately Restricted The Input Space
To Improve "Testability"

- In General, Sensors Are Processed With Too Much Precision

  E.G.: Temperature Stored As Floating Point Quantity

- MSS Examples:
  - User Input Seed Motion Distance Rounded To 1/10th Millimeter
  - User Input Seed Motion Angles Rounded To 1/10th Degree

- Reduces Input Space From "Huge" To ~10^9 Per Possible Seed Position

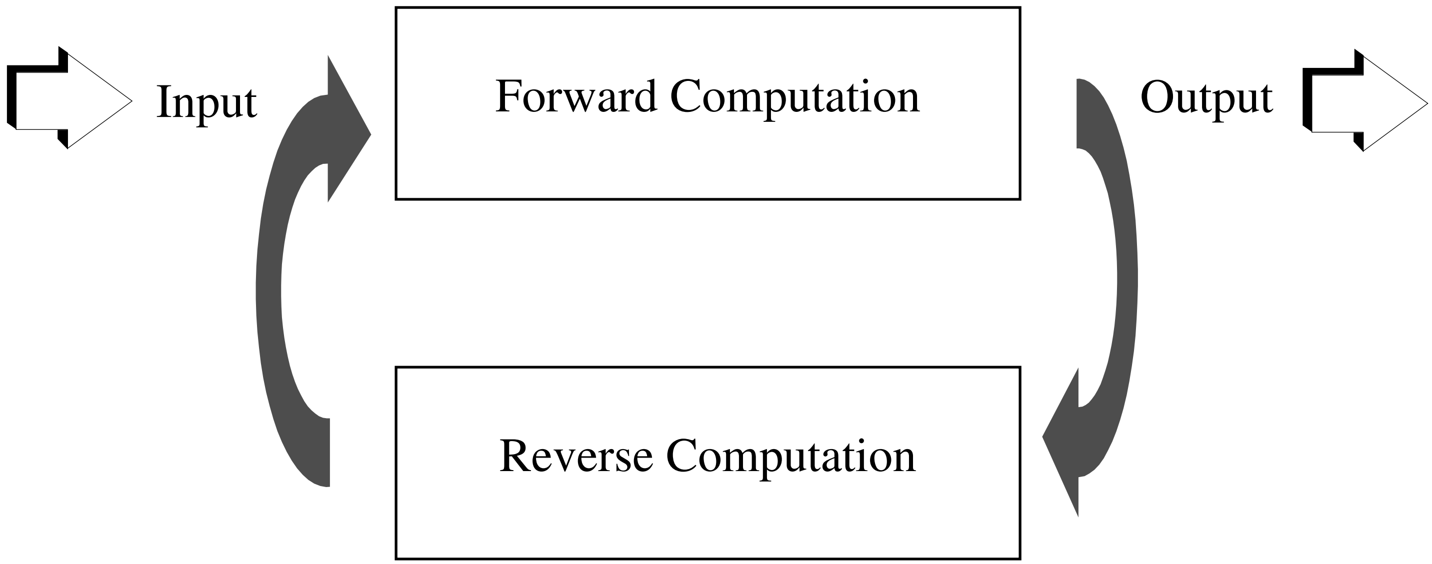
- Has *No* Effect On System Utility

- Permits Significant Statistical Samples If Not Exhaustive

UVA
Department of Computer Science

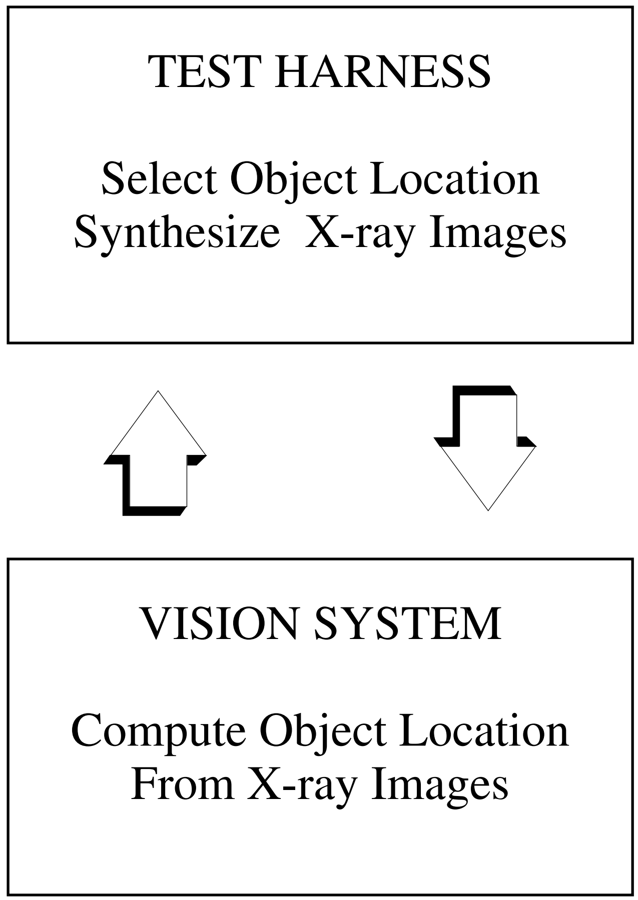
MSS CORRECTNESS CHECKING

- Complicated Computations - How Do We Check Them?
- In This Case, Systematic Application Of Reversal Checks (Or Similar)
- Reversal Check:



UVA
Department of Computer Science

VISION SYSTEM - OBJECT LOCATION



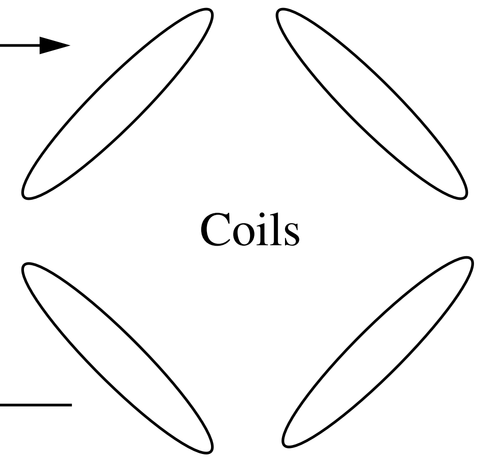
UVA
Department of Computer Science

MAGNETIC FIELD COMPUTATION

Required Force On Seed →

- Current Calculation:
  - Very Complex
  - One To Many
  - Difficult To Optimize

→



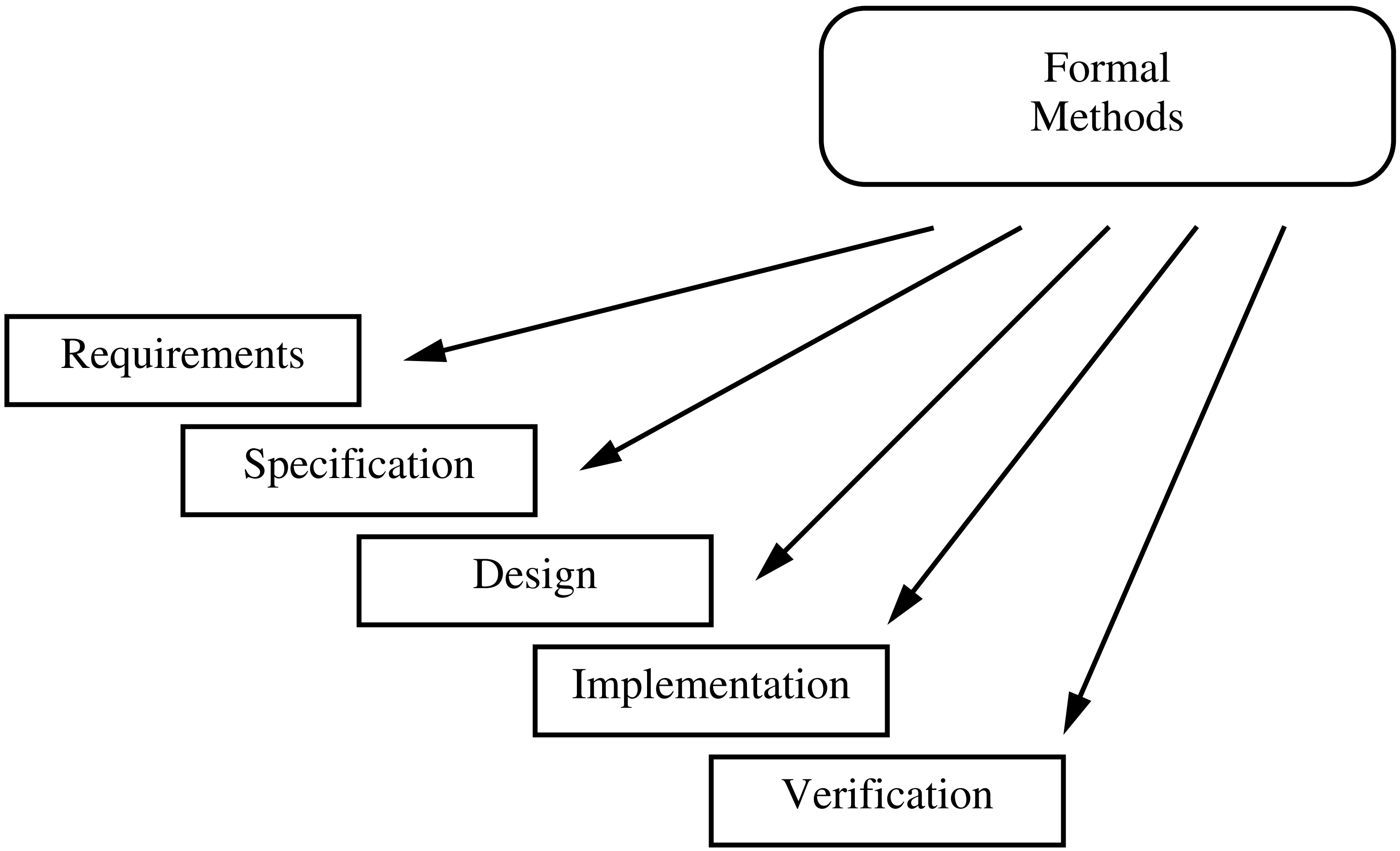
←

- Force Calculation:
  - Very Simple
  - One To One

← ?

UVA
Department of Computer Science

CONCLUSIONS



UVA
Department of Computer Science