N96-10033
59511
p. 9

SAFETY ANALYSIS

John C. Knight
Department of Computer Science, University of Virginia
Charlottesville, VA 22903

Case Studies

We are engaged in a research program in safety-critical computing that is based on two case studies. We use these case studies to provide application-specific details of the various research issues, and as targets for evaluation of research ideas.

The first case study is the *Magnetic Stereotaxis System* (MSS), an investigational device for performing human neurosurgery being developed in a joint effort between the Department of Physics at the University of Virginia and the Department of Neurosurgery at the University of Iowa.

The system operates by manipulating a small permanent magnet (known as a "seed") within the brain using an externally applied magnetic field. By varying the magnitude and gradient of the external magnetic field, the seed can be moved along a non-linear path and positioned at a site requiring therapy, e.g., a tumor. The magnetic field required for movement through brain tissue is extremely high, and is generated by a set of six superconducting magnets located in a housing surrounding the patient's head. The system uses two X-ray cameras positioned at right angles to detect in real time the locations of the seed and of X-ray opaque markers affixed to the patient's skull. The X-ray images are used to locate the objects of interest in a canonical frame of reference.

The second case study is the *University of Virginia Research Nuclear Reactor* (UVAR). It is a 2 MW thermal, concrete-walled pool reactor. The system operates using 20 to 25 plate-type fuel assemblies placed on a rectangular grid plate. There are three scramable safety rods, and one non-scramable regulating rod that can be put in automatic mode. It was originally constructed in 1959 as a 1 MW system, and it was upgraded to 2 MW in 1973. Though only a research reactor rather than a power reactor, the issues raised are significant and can be related to the problems faced by full-scale reactor systems.

Safety Kernel

The software in systems like those in our case studies is very large and complex. We assume that, because of this size and complexity, faults will remain in the software for an application after development. An approach we are pursuing to deal with this is a software architecture termed a *safety kernel*, a concept directly analogous to the security kernel used in security applications.

A *security kernel* provides assurance that a set of *security policies* is enforced independently of the application program. Verification of the security kernel is sufficient to ensure enforcement of those policies encapsulated within the security kernel. The application program need not enforce the security policies, and it can, in fact, undertake actions that would normally lead to violation of the security policies with no danger of actual violations taking place. The similarity between security concerns and safety concerns is considerable and the concept of a safety kernel is appealing. If the concept were feasible, a safety kernel would enforce a set of *safety policies* by monitoring requests to devices, device actions, device status, application software status, and so on.

We have developed an enforcement safety kernel and integrated it into our MSS implementation. The safety kernel is generated automatically from a formal specification of the safety policies, and tests of the MSS instantiation show excellent performance.

Testing

Systems of this complexity pose significant challenges in the area of testing, especially in the large number of possible test cases. We are using a technique that we call *specification limitation* to permit demonstration of useful properties by exhaustive testing. By specification limitation we mean that the specification for the application is deliberately limited in several areas to restrict the total number of test cases. For example, in the MSS the angles entered by the operator for the required direction of motion are rounded to 1/10 of a degree. In practice, this is not a significant functional restriction but it permits exhaustive testing of the angles used for setting direction. The same approach is used with distance.

A second significant problem in testing complex systems is correctness determination, i.e., determining whether the outputs are correct. In our MSS implementation, we have addressed this problem by the use of *reversal checks* on the entire system. A reversal check computes a program's input from its output and compares this with the actual input. The current calculations for the superconducting coils, for example, begin with a required force and are very complex. Computing the force resulting from the coil currents, however, is simple and provides the exact inverse of the current calculations. Thus the input can be computed and compared. A variation on the idea of a reversal check is also used by the MSS imaging subsystem.

SAFETY ANALYSIS*

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

John C. Knight

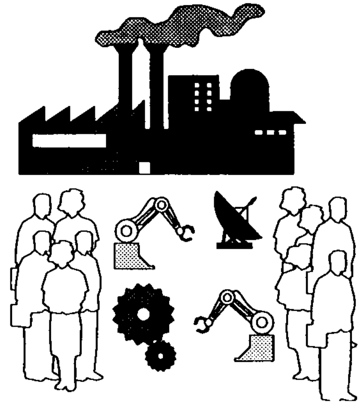
Department of Computer Science
University of Virginia
Charlottesville, Virginia 22903

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

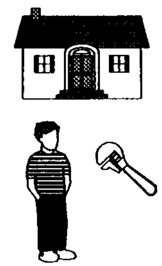
UVA
Department of Computer Science

EXPERIENCE WITH SAFETY-CRITICAL SYSTEMS

Industry/Government

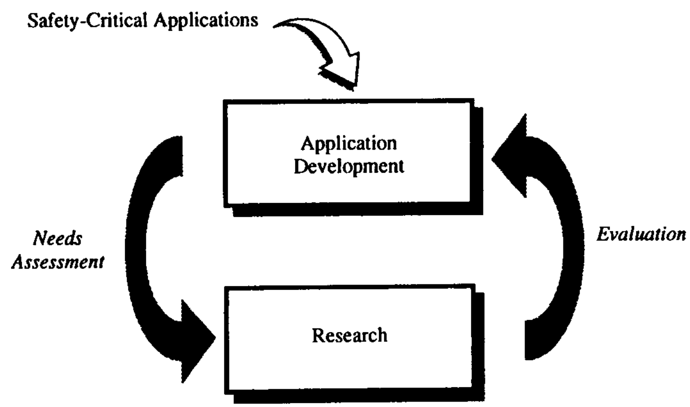


University



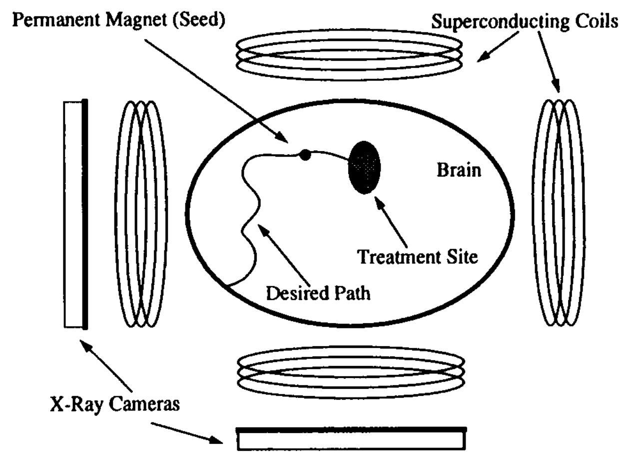
UVA
Department of Computer Science

RESEARCH BASED ON CASE STUDIES



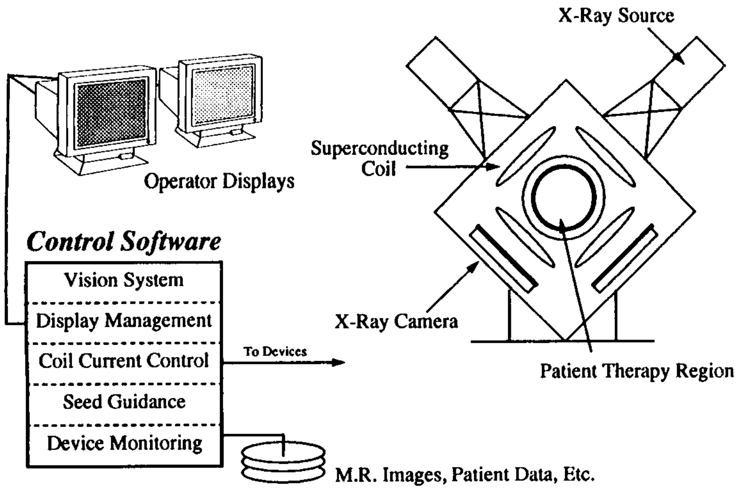
UVA
Department of Computer Science

CASE STUDY MAGNETIC STEREOTAXIS SYSTEM - CONCEPT



UVA
Department of Computer Science

Case Study - Magnetic Stereotaxis System



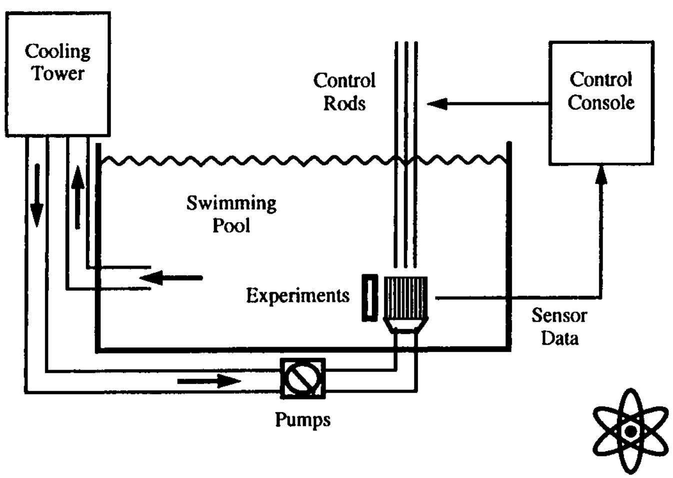
UVA
Department of Computer Science

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

UVA
Department of Computer Science

Case Study - UVa Research Nuclear Reactor



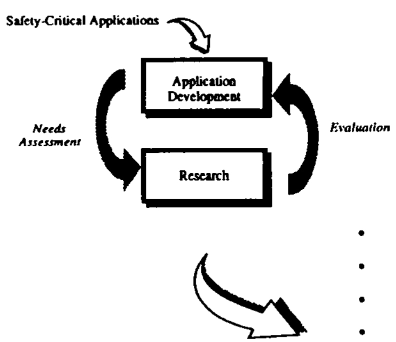
UVA
Department of Computer Science

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

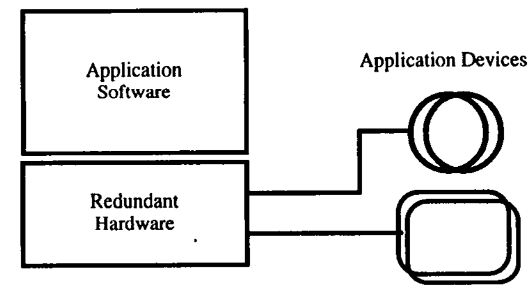
Research Topics



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

UVA
Department of Computer Science

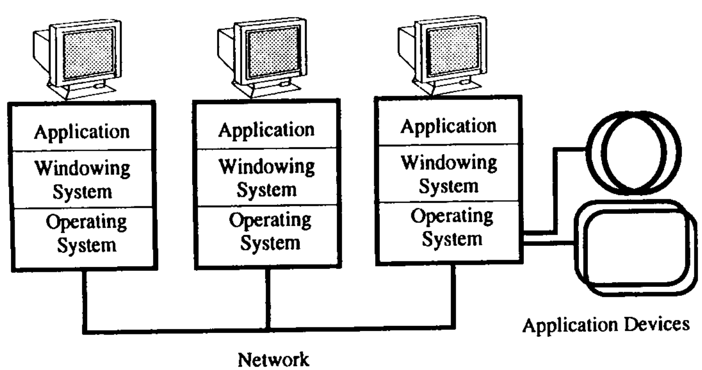
Naive Application Architecture



- Keep It Simple, S*****

UVA
Department of Computer Science

Realistic Application Architecture

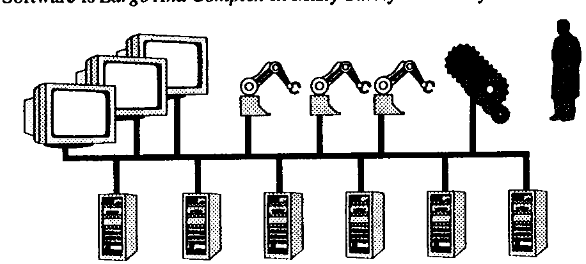


- Diverse Hardware, Network, High-performance Displays
- Extensive, Diverse And Unreliable Software, Perhaps Off-the-shelf

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

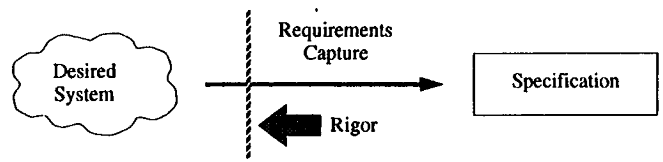
Clearly, To Build A System, We Need Some Form Of Formal Specification.

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...

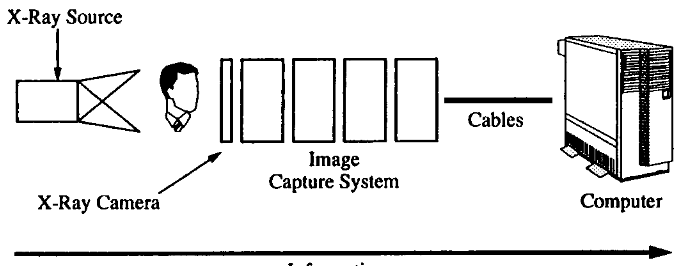
REQUIREMENTS CAPTURE - PROCESS GOALS

- 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



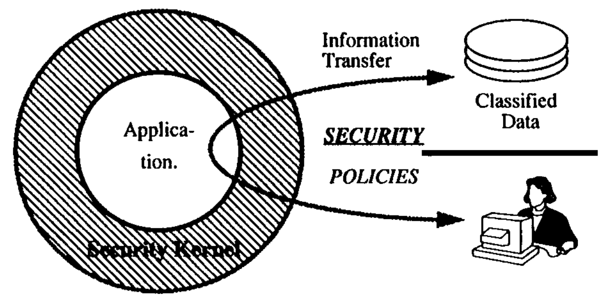
INFORMATION-FLOW MODELLING

- 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



SECURITY KERNEL CONCEPT

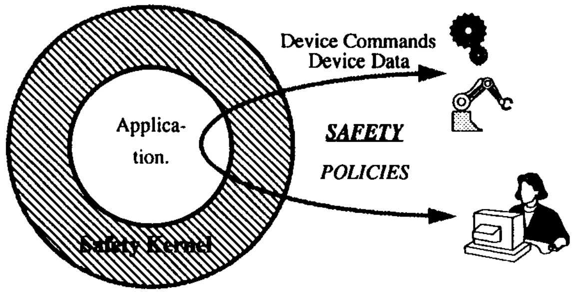
- Concept Is That Security Kernel Controls Access To All Information
- Kernel Enforces A Set Of *Security* Policies Irrespective Of Application Software's Actions:



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

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

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.....

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.....

Support vs. Enforcement Kernels

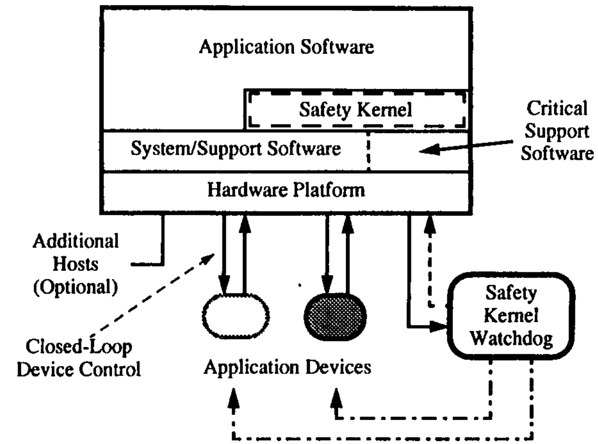
SUPPORT **KERNEL:**	***ENFORCEMENT*** **KERNEL:**
• Support provided for policy enforcement	• Enforces selected policies
• Application software responsible for policy enforcement	• Application software is not responsible for kernel-enforced policies
• Enforcement depends on correct use (or use at all) of kernel operations	• Policies enforced irrespective of use of kernel operations
• Support provided as potentially useful services, e.g., assertions, timers, device drivers	• Kernel not traditional - not necessarily implemented on bare hardware
• Examples: traditional operating systems, some earlier safety kernels	• Kernel enforces policies much like a fuse enforces a policy in an electrical system

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?

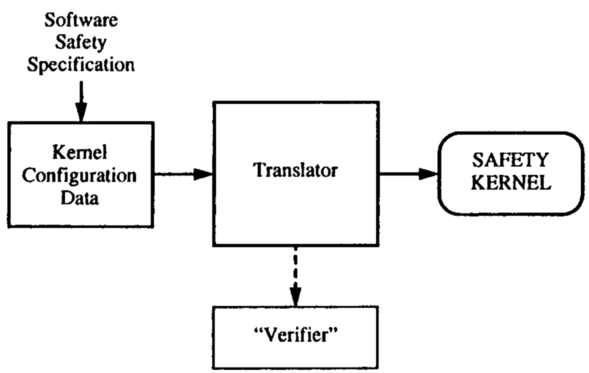
UVA
Department of Computer Science

SYSTEM DESIGN



UVA
Department of Computer Science

SAFETY-KERNEL IMPLEMENTATION



UVA
Department of Computer Science

TESTING ISSUES

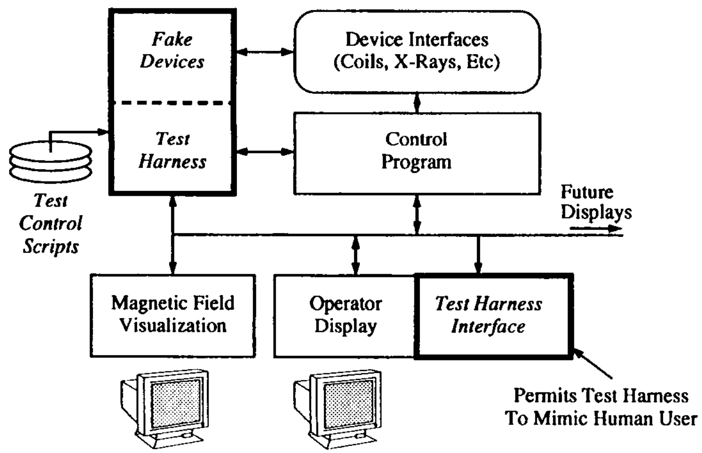
- These Systems:
  - Have Significant Graphic User Interfaces
  - Involve Complex Mathematics
  - Have Stringent Safety Requirements
- Important Research Questions:
  - How Can They Be Tested? They Are Very Complex Systems
  - What Would Testing Mean? We Would Like Meaningful Results
- Testing Safety-critical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

☞ ***BUT:***

Maybe Restricted But Useful Properties Can Be Formally Established

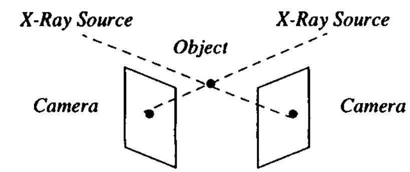
UVA
Department of Computer Science

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"

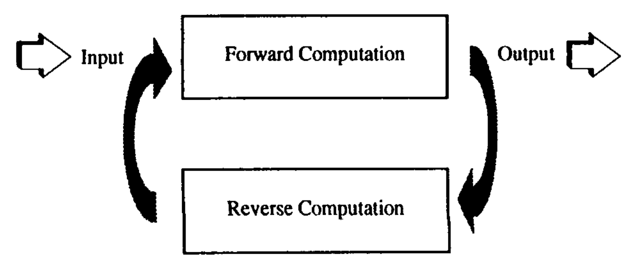
- In General, Sensors Are Processed With Too Much Precision

  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

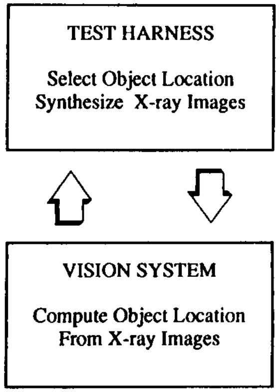
MSS Correctness Checking

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



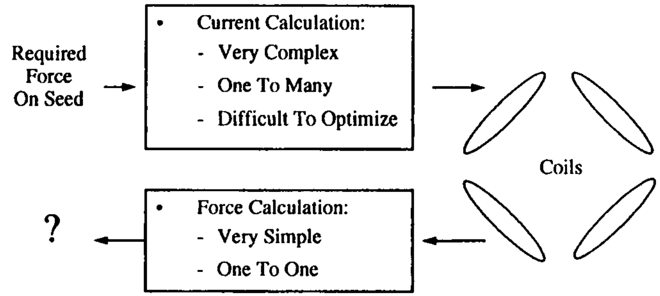
UVA
Department of Computer Science

Vision System - Object Location



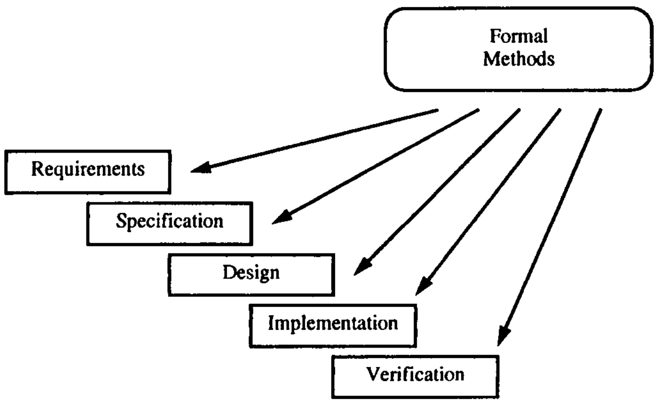
UVA
Department of Computer Science

Magnetic Field Computation



UVA
Department of Computer Science

Conclusions



UVA
Department of Computer Science