



ILLINOIS INSTITUTE
OF TECHNOLOGY

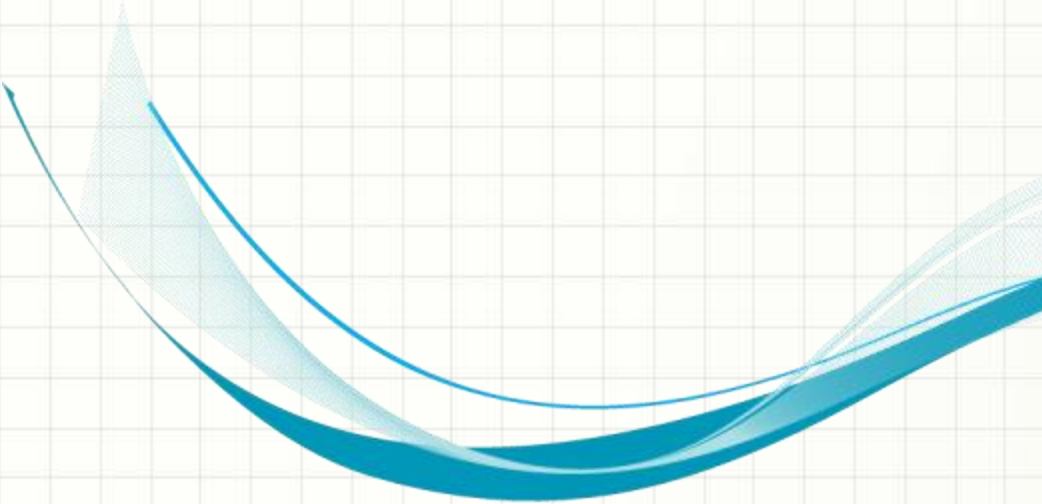
Transforming Lives. Inventing the Future.

www.iit.edu

SOFTWARE ENGINEERING

CS 487

Dennis Hood
Computer Science



Week 7

Artificially Intelligent Systems

Lesson Overview

- Artificially Intelligent Systems
- Reading
 - Ch. 12 – Safety-critical Systems
 - Ch. 13 – Security Engineering
- Objectives
 - Examine the concepts of awareness and intelligence in terms of automation
 - Discuss engineering approaches to achieve artificial awareness and intelligent decision-making
 - Explore the concepts of safety-critical and security, mission-critical non-functional requirements
 - Look at engineering approaches for creating applications with these stringent requirements

Topics for Discussion

- Discuss the implications of a “failure of imagination” in the context of software engineering.
- Discuss the role of automated awareness in exception detection.
- Discuss the role of automated intelligent decision-making in exception handling.
- Discuss risk assessment for safety- and security-critical systems.
- Compare proactive and reactive QA for software engineering to security approaches on the IIT campus.

Human Awareness

- Human detection of exceptional states
 - Similar to computer system exception detection in that the human is assuming normal and must be made to recognize the change to exceptional
 - Therefore a similar protocol must exist which distinguishes the current state as exceptional
 - Of course, this will not be effective if the human is so disengaged as to not “receive” the message
 - Given that one of the fundamental benefits of automation is to free humans from engagement, this disengagement problem is more likely
- Human handling of exceptions
 - The protocol should specify a clear set of actions for the human to take to return to at least “safe” if not normal
 - The protocol must contain acknowledgement-based verification to insure that the human has responded and has taken control of the situation

Automated Exception Management

- Exception detection
 - The first step in solving a problem is recognizing that the problem exists
 - The characteristics of the exceptional state must clearly distinguish it from any other possible state
 - What if the system does not “recognize” the characteristics as exceptional?, or what if the system mis-identifies them as indicating a different kind of exception?
- Exception handling
 - Once the system recognizes that it is in the exceptional state, it must quickly execute handling code to recover
 - If feasible, recovery should be a return to normal
 - Else, recovery can be to an agreed upon “safe” state
 - What if no automated handler has been provided?

Safety-critical Systems

- Safety-critical is a non-functional requirement meaning that system operation must always be in a safe state
- Primary safety-critical software
 - Embedded controllers where a failure can result in a hardware malfunction resulting in human injury or environmental damage
- Secondary safety-critical software
 - Software which, in the event of failure, can result in injury
 - For example, a defective computer-aided design tool which produces a flawed design

Hazard-driven Analysis

- Hazard identification
 - Imagine the hazards that may threaten the system
- Hazard assessment
 - Prioritize identified hazards (low priority hazards may not “justify” further engineering effort)
- Hazard analysis
 - Use root-cause analysis to map the early events which lead to failure
- Risk reduction
 - Identify factors which can reduce the likelihood and/or impact of hazards

Requirements Drive the Design

- Much of design is choosing the best solution by evaluating the degree to which each possible design satisfies the prioritized non-functional requirements
- Evaluations with respect to safety-criticality
 - Hazard avoidance – the system is designed to avoid hazards
 - Hazard detection and removal – the system is designed to detect problems and correct them before an accident occurs
 - Damage limitation – the system is designed to minimize the impact of a problem when it occurs

Safety Engineering Processes

- Safety Assurance
 - Defining and executing the activities which assure that the system will operate safely
- Formal Verification
 - Establishment of formal methods for “proving” proper operation
- Model Checking
 - Modeling system state behavior and establishing (often tool-based) assessment of the model
- Static Program Analysis
 - Establishment of (often tool-based) assessments of code to “discover” possible faults and anomalies

Designing for Security

- Base decisions on an explicit security policy
- Use defense in depth by employing multiple layers of security
- Fail securely so that failure does not result in exposure
- Balance security and usability (e.g., PINs)
- Log user actions
- Use redundancy and diversity to reduce risk (e.g., maintain backups, avoid reliance on single possibly vulnerable platforms)
- Specify and restrict the format of system inputs
- Compartmentalize assets (user's "need to know")
- Design for deployment with proper and secure configuration