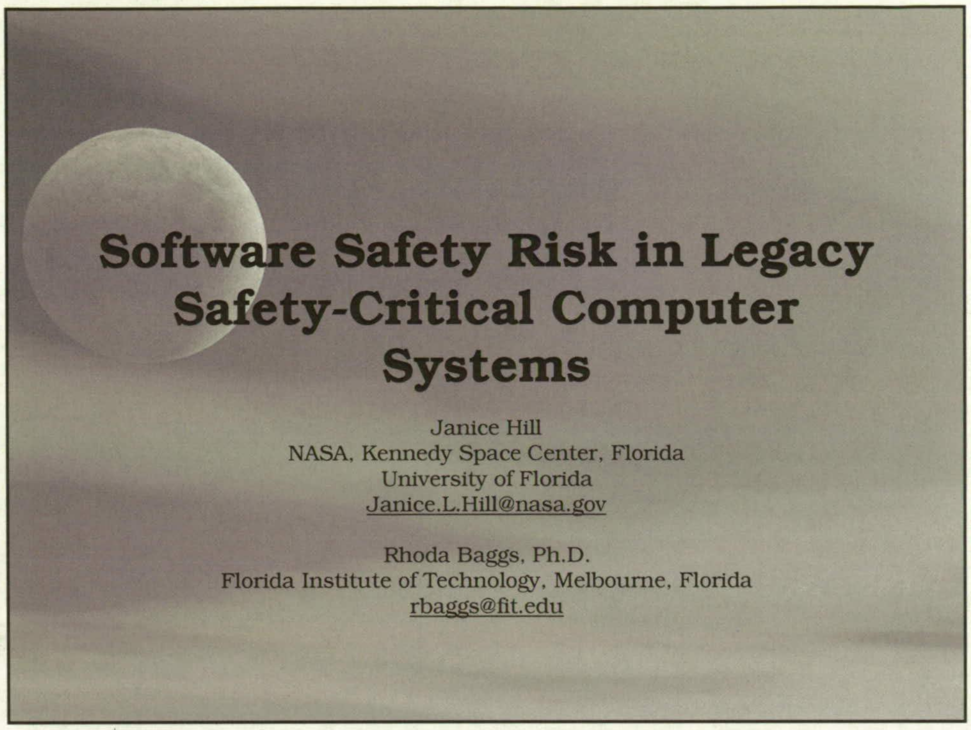
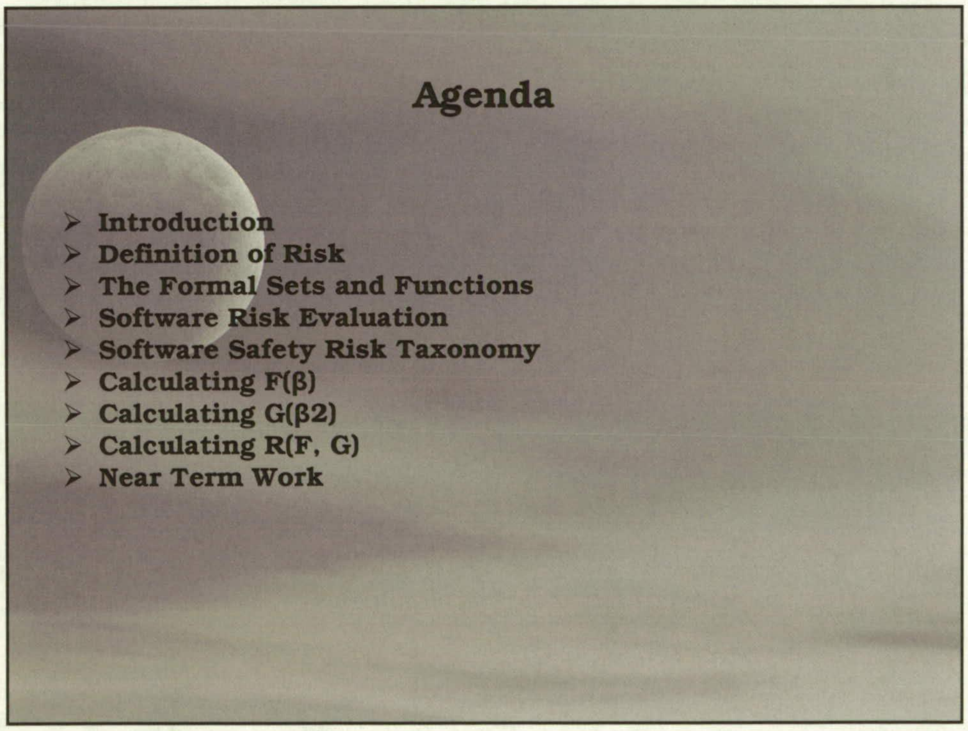


Software Safety Risk in Legacy Safety-Critical Computer Systems

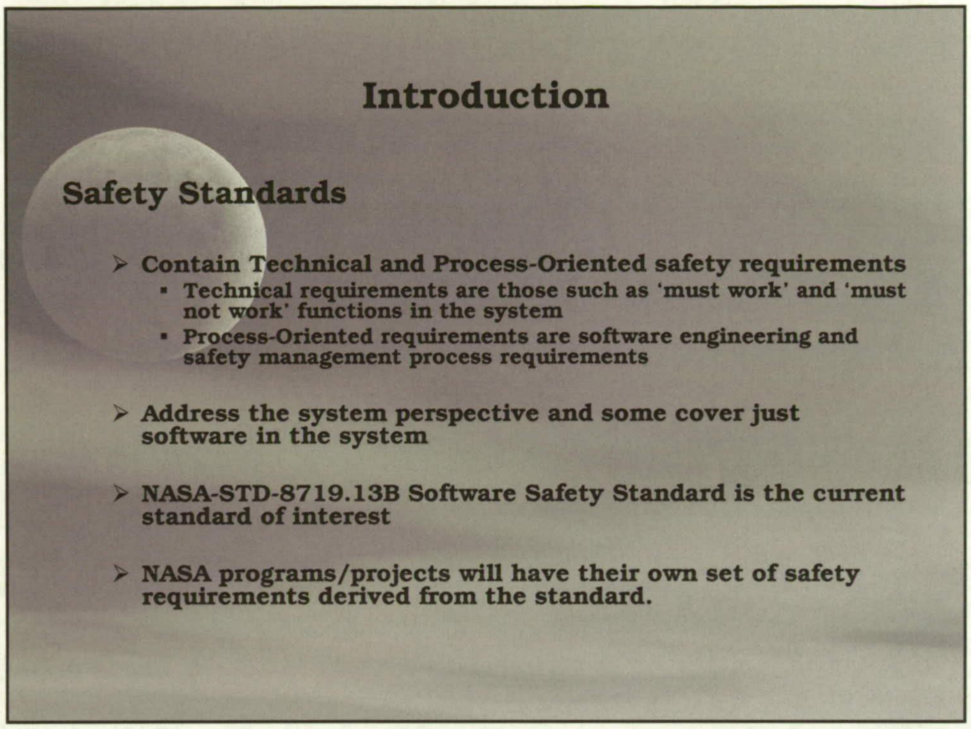
Janice Hill
NASA, Kennedy Space Center, Florida
University of Florida
Janice.L.Hill@nasa.gov

Rhoda Baggs, Ph.D.
Florida Institute of Technology, Melbourne, Florida
rbaggs@fit.edu



Agenda

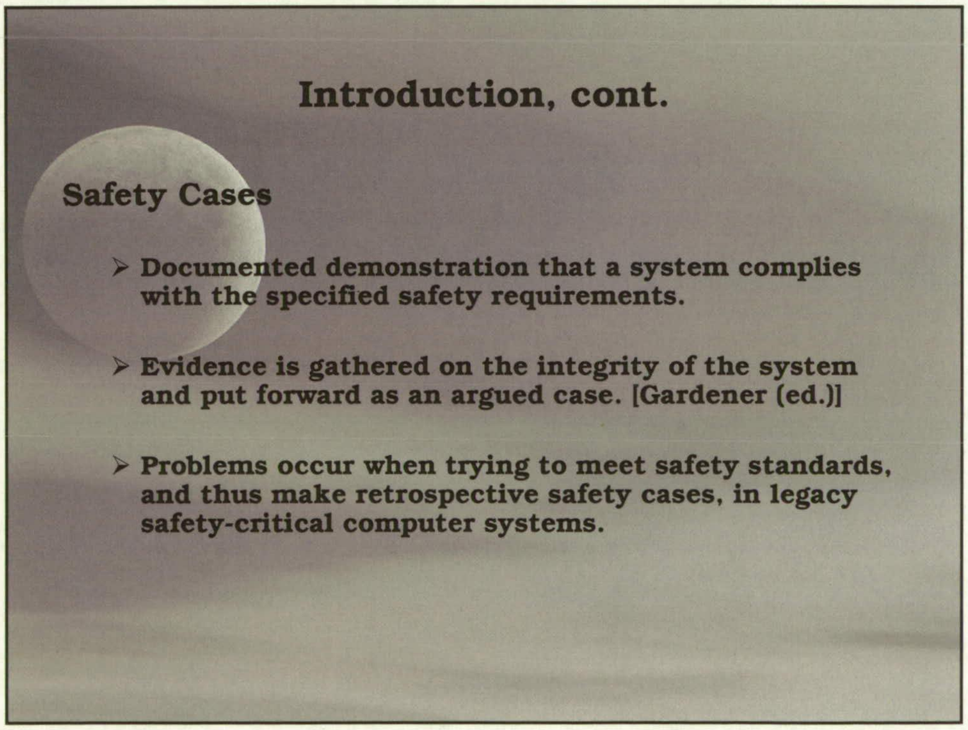
- Introduction
- Definition of Risk
- The Formal Sets and Functions
- Software Risk Evaluation
- Software Safety Risk Taxonomy
- Calculating F(β)
- Calculating G(β2)
- Calculating R(F, G)
- Near Term Work



Introduction

Safety Standards

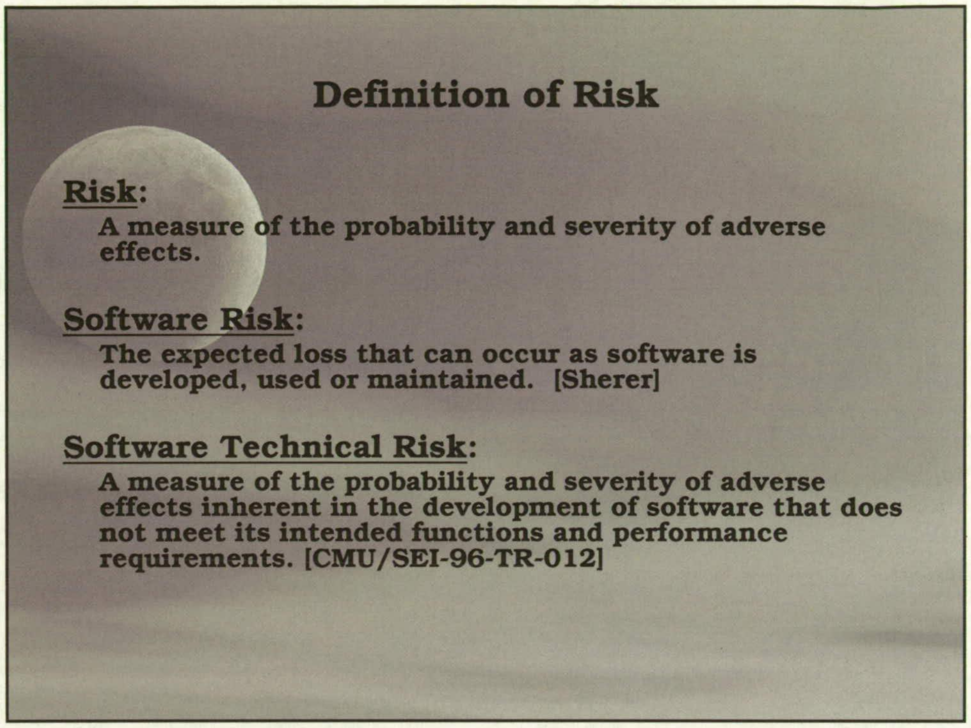
- Contain Technical and Process-Oriented safety requirements
  - Technical requirements are those such as 'must work' and 'must not work' functions in the system
  - Process-Oriented requirements are software engineering and safety management process requirements
- Address the system perspective and some cover just software in the system
- NASA-STD-8719.13B Software Safety Standard is the current standard of interest
- NASA programs/projects will have their own set of safety requirements derived from the standard.



Introduction, cont.

Safety Cases

- Documented demonstration that a system complies with the specified safety requirements.
- Evidence is gathered on the integrity of the system and put forward as an argued case. [Gardener (ed.)]
- Problems occur when trying to meet safety standards, and thus make retrospective safety cases, in legacy safety-critical computer systems.

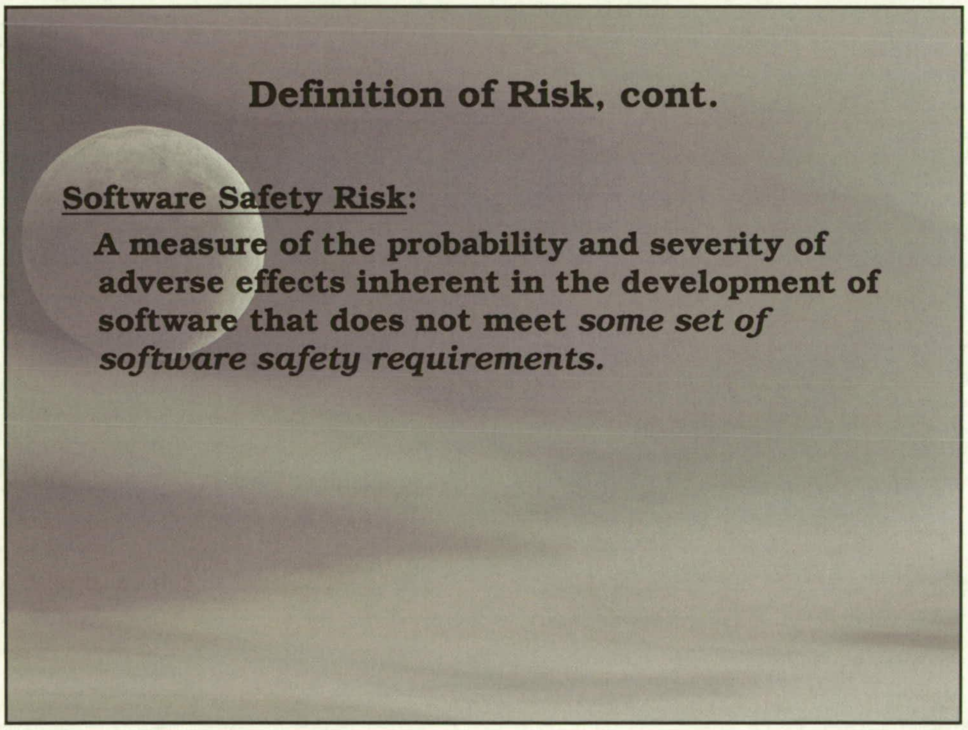


Definition of Risk

Risk:
A measure of the probability and severity of adverse effects.

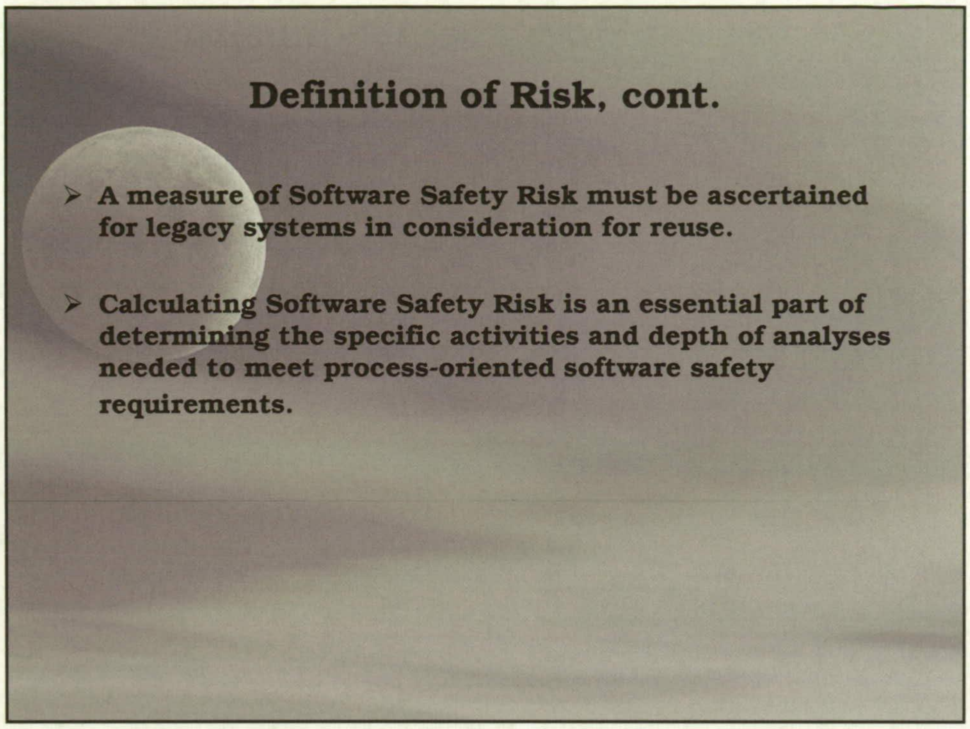
Software Risk:
The expected loss that can occur as software is developed, used or maintained. [Sherer]

Software Technical Risk:
A measure of the probability and severity of adverse effects inherent in the development of software that does not meet its intended functions and performance requirements. [CMU/SEI-96-TR-012]



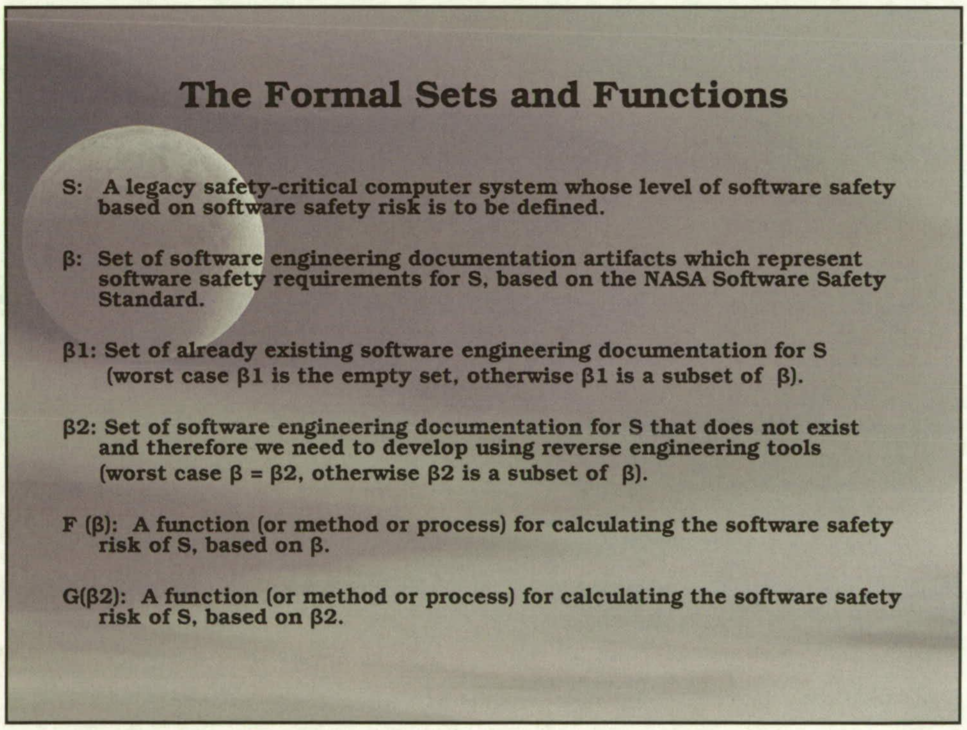
Definition of Risk, cont.

Software Safety Risk:
A measure of the probability and severity of adverse effects inherent in the development of software that does not meet *some set of software safety requirements.*



Definition of Risk, cont.

- A measure of Software Safety Risk must be ascertained for legacy systems in consideration for reuse.
- Calculating Software Safety Risk is an essential part of determining the specific activities and depth of analyses needed to meet process-oriented software safety requirements.



The Formal Sets and Functions

S: A legacy safety-critical computer system whose level of software safety based on software safety risk is to be defined.

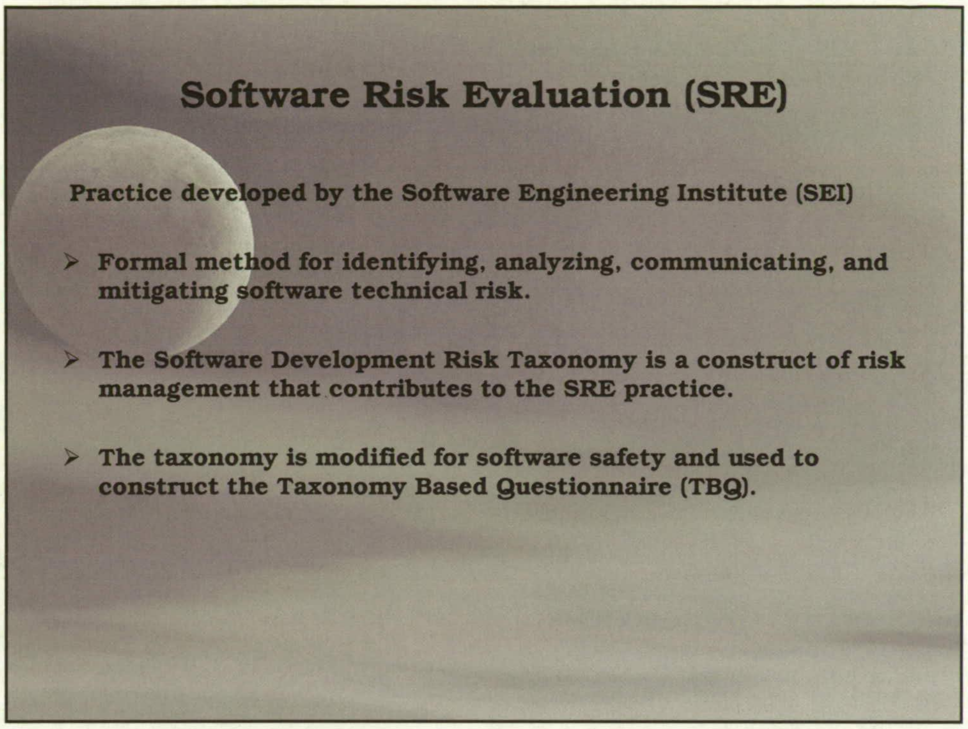
β: Set of software engineering documentation artifacts which represent software safety requirements for S, based on the NASA Software Safety Standard.

β1: Set of already existing software engineering documentation for S (worst case β1 is the empty set, otherwise β1 is a subset of β).

β2: Set of software engineering documentation for S that does not exist and therefore we need to develop using reverse engineering tools (worst case β = β2, otherwise β2 is a subset of β).

F (β): A function (or method or process) for calculating the software safety risk of S, based on β.

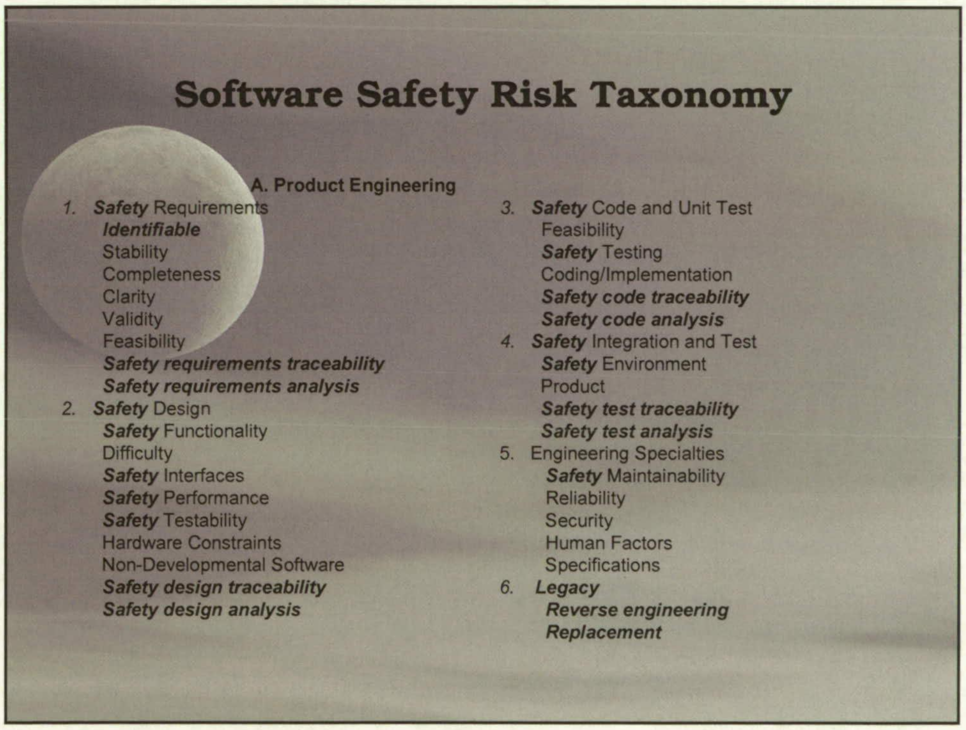
G(β2): A function (or method or process) for calculating the software safety risk of S, based on β2.



Software Risk Evaluation (SRE)

Practice developed by the Software Engineering Institute (SEI)

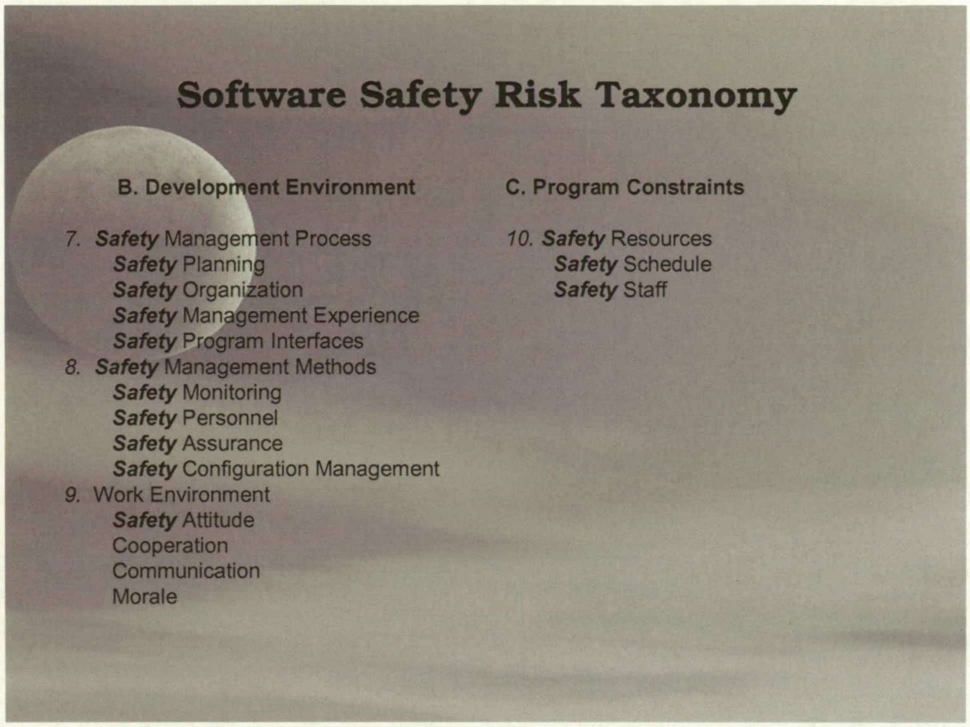
- **Formal method for identifying, analyzing, communicating, and mitigating software technical risk.**
- **The Software Development Risk Taxonomy is a construct of risk management that contributes to the SRE practice.**
- **The taxonomy is modified for software safety and used to construct the Taxonomy Based Questionnaire (TBQ).**



Software Safety Risk Taxonomy

A. Product Engineering

1. ***Safety*** Requirements
   - ***Identifiable***
   - Stability
   - Completeness
   - Clarity
   - Validity
   - Feasibility
   - ***Safety requirements traceability***
   - ***Safety requirements analysis***
2. ***Safety*** Design
   - ***Safety*** Functionality
   - Difficulty
   - ***Safety*** Interfaces
   - ***Safety*** Performance
   - ***Safety*** Testability
   - Hardware Constraints
   - Non-Developmental Software
   - ***Safety design traceability***
   - ***Safety design analysis***
3. ***Safety*** Code and Unit Test
   - Feasibility
   - ***Safety*** Testing
   - Coding/Implementation
   - ***Safety code traceability***
   - ***Safety code analysis***
4. ***Safety*** Integration and Test
   - ***Safety*** Environment
   - Product
   - ***Safety test traceability***
   - ***Safety test analysis***
5. Engineering Specialties
   - ***Safety*** Maintainability
   - Reliability
   - Security
   - Human Factors
   - Specifications
6. ***Legacy***
   - ***Reverse engineering***
   - ***Replacement***



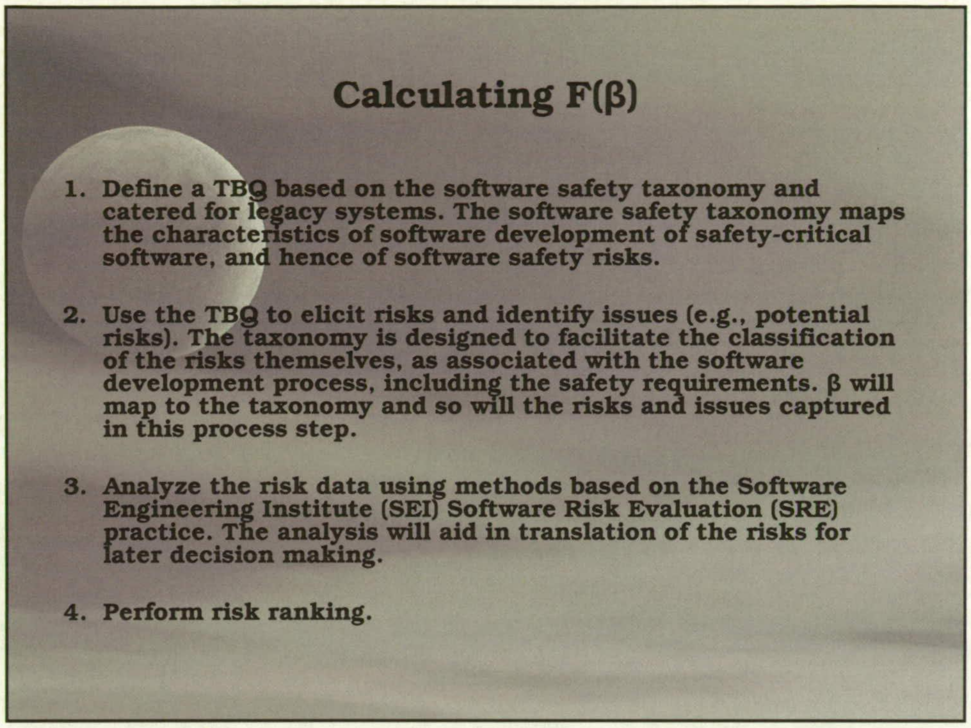
Software Safety Risk Taxonomy

B. Development Environment

7. ***Safety*** Management Process
   - ***Safety*** Planning
   - ***Safety*** Organization
   - ***Safety*** Management Experience
   - ***Safety*** Program Interfaces
8. ***Safety*** Management Methods
   - ***Safety*** Monitoring
   - ***Safety*** Personnel
   - ***Safety*** Assurance
   - ***Safety*** Configuration Management
9. Work Environment
   - ***Safety*** Attitude
   - Cooperation
   - Communication
   - Morale

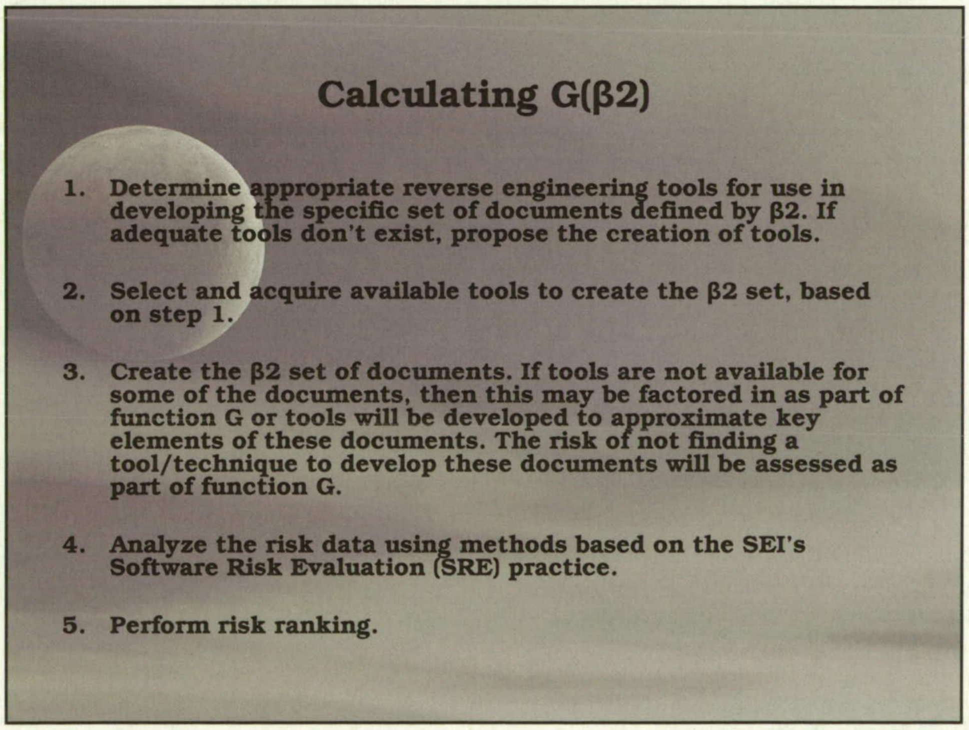
C. Program Constraints

10. ***Safety*** Resources
    - ***Safety*** Schedule
    - ***Safety*** Staff



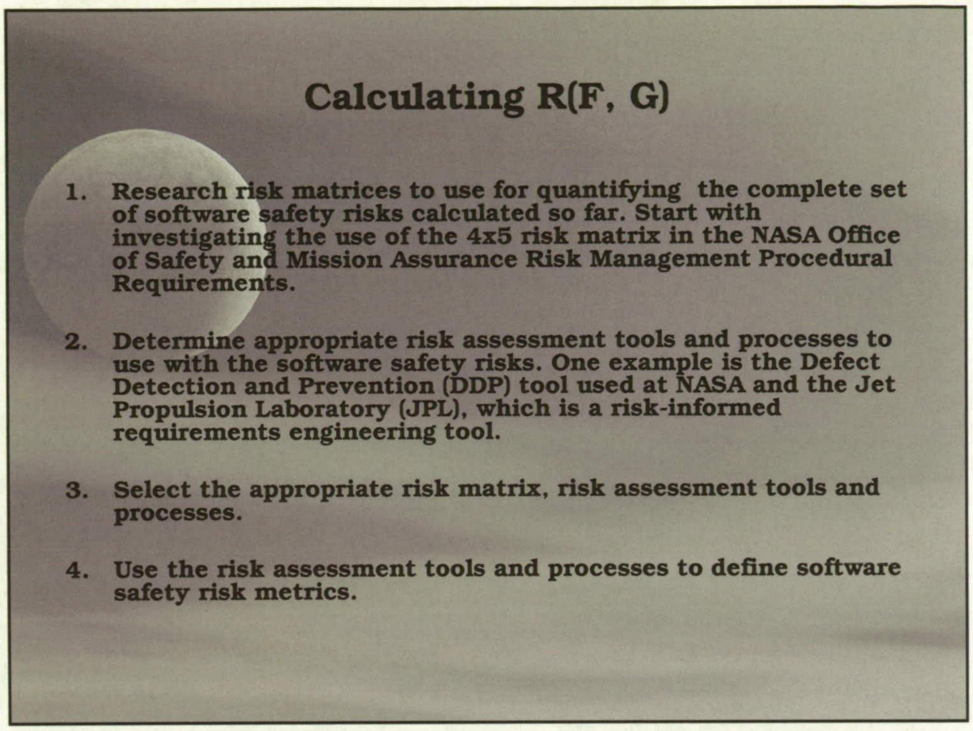
Calculating F(β)

1. Define a TBQ based on the software safety taxonomy and catered for legacy systems. The software safety taxonomy maps the characteristics of software development of safety-critical software, and hence of software safety risks.
2. Use the TBQ to elicit risks and identify issues (e.g., potential risks). The taxonomy is designed to facilitate the classification of the risks themselves, as associated with the software development process, including the safety requirements. β will map to the taxonomy and so will the risks and issues captured in this process step.
3. Analyze the risk data using methods based on the Software Engineering Institute (SEI) Software Risk Evaluation (SRE) practice. The analysis will aid in translation of the risks for later decision making.
4. Perform risk ranking.



Calculating G(β2)

1. Determine appropriate reverse engineering tools for use in developing the specific set of documents defined by β2. If adequate tools don't exist, propose the creation of tools.
2. Select and acquire available tools to create the β2 set, based on step 1.
3. Create the β2 set of documents. If tools are not available for some of the documents, then this may be factored in as part of function G or tools will be developed to approximate key elements of these documents. The risk of not finding a tool/technique to develop these documents will be assessed as part of function G.
4. Analyze the risk data using methods based on the SEI's Software Risk Evaluation (SRE) practice.
5. Perform risk ranking.



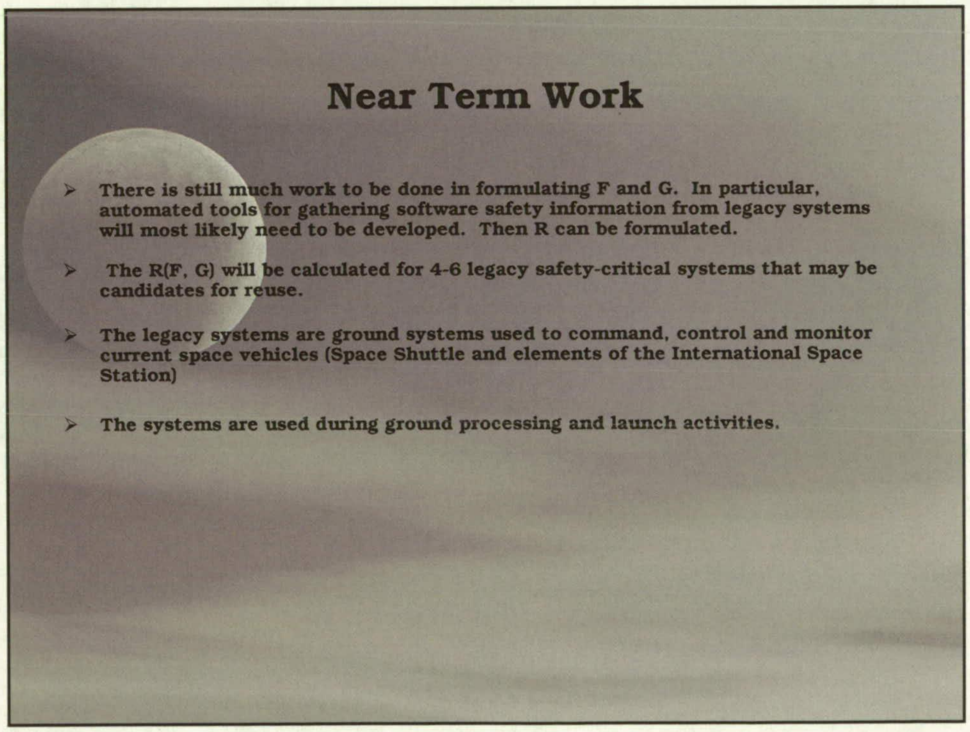
Calculating R(F, G)

1. Research risk matrices to use for quantifying the complete set of software safety risks calculated so far. Start with investigating the use of the 4x5 risk matrix in the NASA Office of Safety and Mission Assurance Risk Management Procedural Requirements.

2. Determine appropriate risk assessment tools and processes to use with the software safety risks. One example is the Defect Detection and Prevention (DDP) tool used at NASA and the Jet Propulsion Laboratory (JPL), which is a risk-informed requirements engineering tool.

3. Select the appropriate risk matrix, risk assessment tools and processes.

4. Use the risk assessment tools and processes to define software safety risk metrics.



Near Term Work

- There is still much work to be done in formulating F and G. In particular, automated tools for gathering software safety information from legacy systems will most likely need to be developed. Then R can be formulated.

- The R(F, G) will be calculated for 4-6 legacy safety-critical systems that may be candidates for reuse.

- The legacy systems are ground systems used to command, control and monitor current space vehicles (Space Shuttle and elements of the International Space Station)

- The systems are used during ground processing and launch activities.