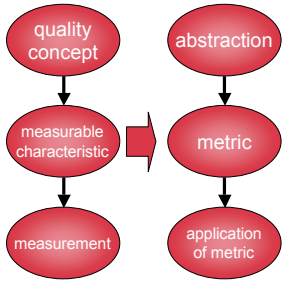
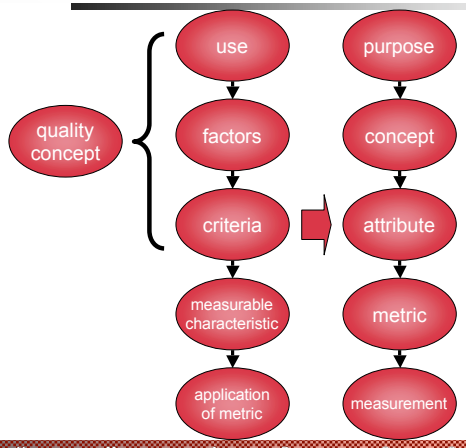
03-Quality & Notation

- Readings:
  - **Fundamentals of Software Engineering**, Ch. 2, 5, 7
  - [[rW98] Roel Wieringa, "A survey of structured and object-oriented software specification methods and techniques," **ACM Computing Surveys** December 1998

Quality



Quality - expanded



Need for metrics

- most "ilities" are qualitative, not quantitative
- how to determine
  - measurement ⇔ experimental
  - analysis ⇔ theoretical
  - simulation ⇔ compute/model

Software Qualities

- External product qualities
  - visible to the user of the system
  - reliability, robustness, performance (efficiency, usability, user-friendliness (human factors)), scalability
- Internal product qualities
  - affect the developers and maintainers
  - correctness (verifiability), maintainability (extensibility, repairability, reusability) portability (understandability, interoperability)

Software Qualities

- Process qualities
  - affect activities
  - productivity, timeliness, visibility

Process qualities ⇒ Internal qualities ⇒ External qualities

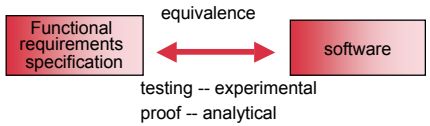
External Product Software Qualities

- reliability
  - "performs as expected" "depend on it"
  - too often release products with known "bugs"
  - SDI arguments
    - can't build correct system that complex -- Dave Parnas
    - doesn't have to be correct to be reliable -- Danny Cohen

most manufacturing ⇒ responsible for reliability
software industry ⇒ waive responsibility

External Product Software Qualities

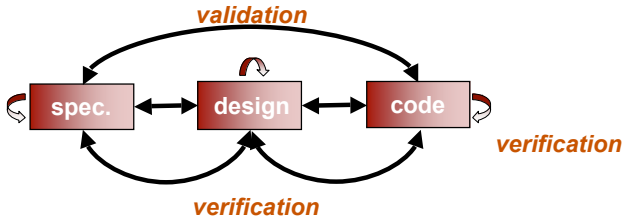
- robustness
  - performs "satisfactorily" even when environment & events unanticipated
- correctness



Reliability, Robustness & Correctness

- correct
  - does exactly what it is defined/specified to do
- reliable
  - does exactly what the user wants (expects?) it to do (under "normal" conditions)
- robust
  - does exactly what the user wants (expects?) it to do (under "abnormal" conditions)
- can apply to products and processes

Correctness as an Internal Product Software Quality



Verifiability

Verification	⇔ Correctness
Validation	⇔ Reliability (+ safety, security, etc.)
Certification	⇔ Legal & Contractual

More External Product SW Qualities

- Performance
  - efficient
    - produces results in an acceptable amount of time
  - usable -- end uses find it easy to use
    - easy to learn
    - easy to install
    - easy to operate
    - easy to advance

More External Product SW Qualities

- User-friendly
  - all of the above is provided with an easy to user interface
- Scalabilty
  - handle expansion in the parameters of the application

Internal Product Software Qualities

- maintainability = repairability + evolvability
  - can be modified and revalidated easily
  - enhanced by abstraction, modularity, discipline, standards, and good taste
  - maintenance is 60% of the lifecycle costs
- understandability
  - some products are inherently more complex than others

Internal Product Software Qualities

- Reusability
  - can be used to construct another product
  - need to plan for reuse
  - can involve any artifact or process
  - difficult
- Reusability factors
  - modularity
  - granularity (e.g., Unix, X windows)
  - trend is for plug-and-play components

Internal Product Software Qualities

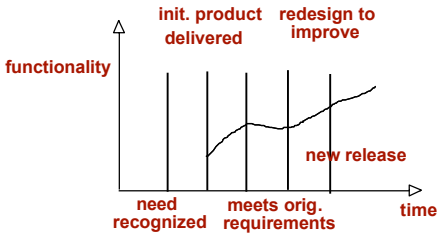
- Interoperability
  - can co-exist and cooperate with other systems
  - easy to integrate
    - open system & layered architectures
    - well-defined, standard interfaces
    - collection of independently written applications that cooperate and function as an integrated system
- Portability
  - can run on different environments (hardware or software platform) with little effort
  - enhanced by using only standard capabilities whenever possible and by isolating non-standard capabilities

Process Qualities

- Productivity
  - measures the performance of the development and maintenance activities
- Visibility
  - allows access to status of both the process and products
    - facilitates management
    - facilitates teamwork

Process Qualities

- Timeliness -- measures the ability to deliver software on time



High-level Goals of SE

- improve productivity
- reduce resources
  - e.g., time, cost, personnel
- improve predictability
- improve maintainability
- improve quality

Process

- Need a process for:
  - Order of activities
  - Product delivery (what, when)
  - Assignment to developers
  - Monitoring ⇒ Measuring ⇒ Planning
- Cannot be (easily) codified or standardized
- Iterative and incremental

Software Processes

- What do people want to do with (to) a product?
  - Find out what it does (quickly, easily):
    UNDERSTANDING
  - Get it to do what is needed (quickly, easily):
    USAGE
  - Not worry about it:
    UNDERSTANDING, EVALUATION, STRESS TESTING
  - Build it (quickly, easily, at low risk):
    DEVELOPMENT
  - Change it as needed (quickly, easily):
    MODIFICATION
  - Improve it (quickly, easily):
    EVOLUTION

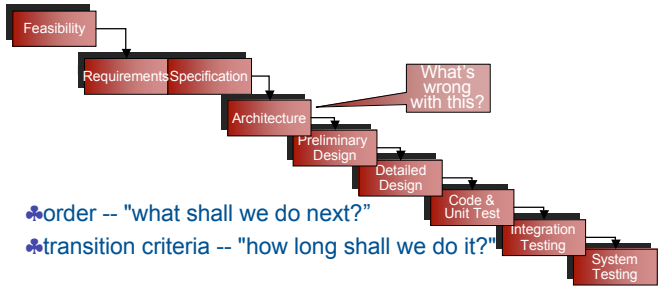
What are the key (sub-)processes it must support

Process

- Current Strong Emphasis on Process
  - Process and Product complement each other
- More past focus on product
- Process focus has been less technical
  - More managerial

Early waterfall model

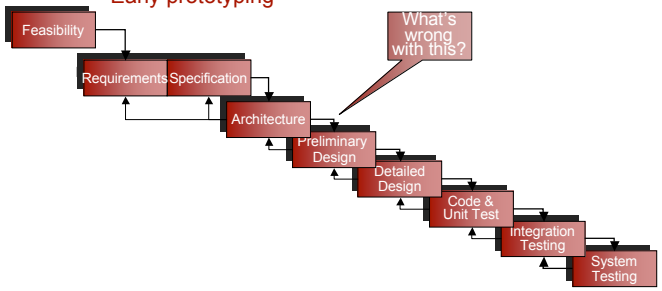
- Earliest design and test



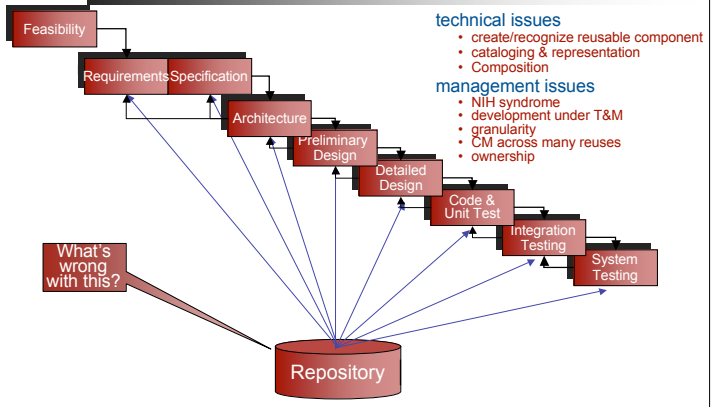
- ♣order -- "what shall we do next?"
- ♣transition criteria -- "how long shall we do it?"

1970's

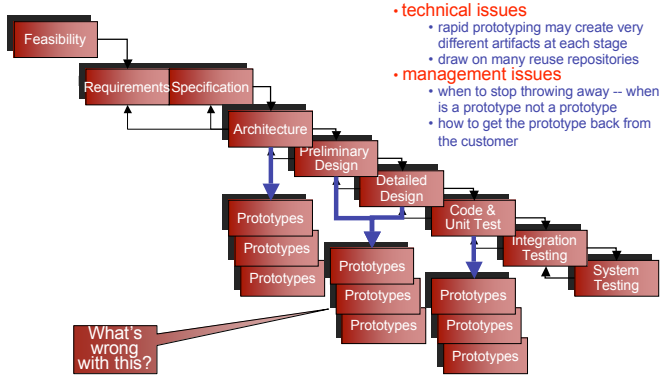
- Recognition of feedback loops
  - Confined to successive stages
- "Build it twice"
  - Early prototyping



Reuse model

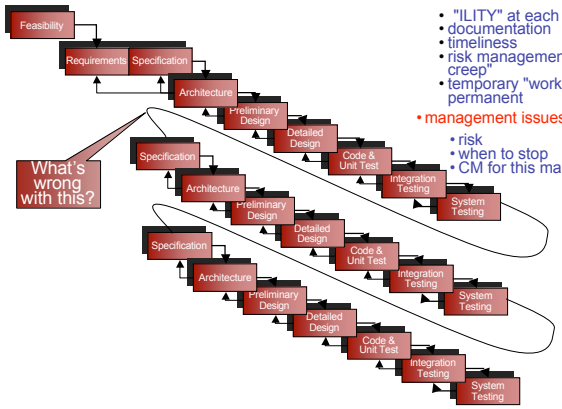


"Throwaway" prototyping



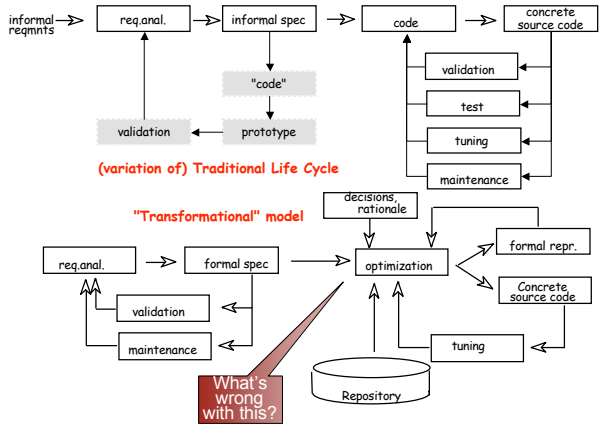
- technical issues
  - rapid prototyping may create very different artifacts at each stage
  - draw on many reuse repositories
- management issues
  - when to stop throwing away -- when is a prototype not a prototype
  - how to get the prototype back from the customer

Evolutionary prototyping

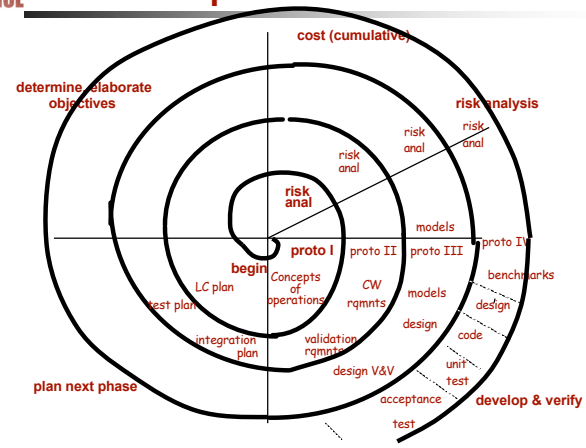


- technical issues
  - "ILITY" at each level
  - documentation
  - timeliness
  - risk management, e.g., "feature creep"
  - temporary "work arounds" become permanent
- management issues
  - risk
  - when to stop
  - CM for this many artifacts

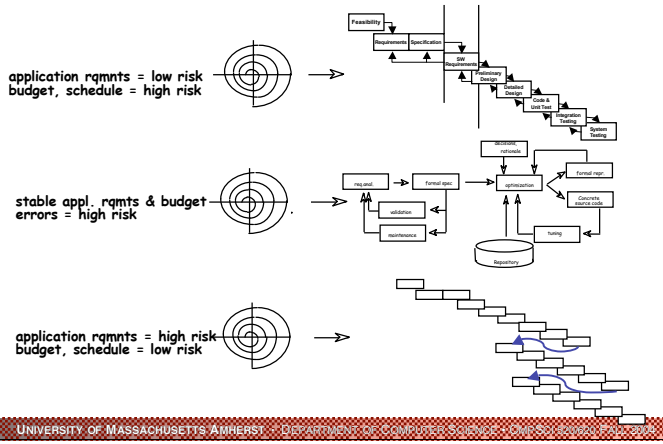
Transformational Model



Boehm's Spiral Model

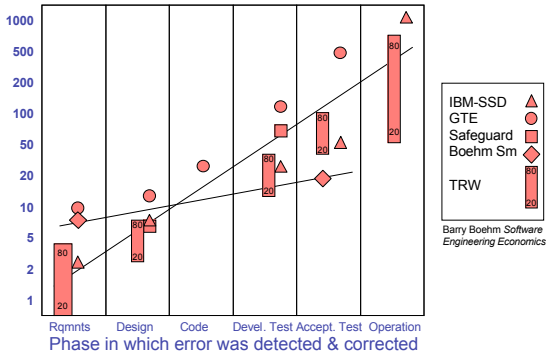


Boehm's Spiral Model



Increase in cost

- to fix or change throughout lifecycle



Back to Stakes & Stakeholders

- Customer:
  - Easy to get it to do what it is supposed to do
  - Cost (time to learn, ease of use) is consistent with benefits
- End User:
  - Does what it is supposed to do
  - Easy to learn, use, support and adapt (as customers' needs change)
  - Maintains job security
- Developer:
  - It is always "under control", and easy to modify
  - It **clearly** does what it is supposed to do
- Management
  - Progress is visible and satisfactory
  - Doesn't cost too much, or more than expected
  - Improves and optimizes business practices
- Innocent Bystander:
  - It does no harm

One Major Focus of SE

- How to manage the creation and maintenance of a software product that satisfies all needs of all stakeholders?
  - Implies understanding who stakeholders are; what questions they need answered; to what degree of thoroughness
  - Implies supporting reasoning needed to provide adequate answers to stakeholder questions (at acceptable cost)
  - Implies need for representation(s) of product sufficient to support such reasoning; and reasoning techniques

Answering Stakeholder Questions

- Define what it means for product to be "correct"
- Ideally
  - Specified early in project
  - Created incrementally as product is built
  - Keep stakeholders satisfied continually
- Some examples
  - Test results must be consistent with expectations (ie. The manual is right)
  - Executables must correspond to source
  - Code must implement design
  - Lower levels of design must be elaborations of higher
  - Test data must really represent expected usage
  - Proofs of concepts really connect proofs to concepts

What do Stakeholders Want to Know?

- Some examples of understandings needed:
  - What does the product do and how do we know
  - What is the product supposed to do
  - How does it work
  - What would happen if I did .....
  - Suppose we change ....
- There are infinitely many such questions
  - For each there are endless varieties of answers
  - The answers themselves form key parts of the product
- Superior products are tightly interconnected bundles of component artifacts, used as the basis for:
  - These questions
  - Their answers
  - Solid basis for believing the answers

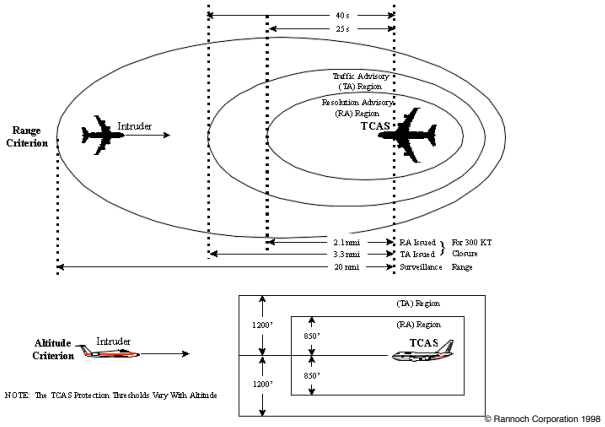
A Useful (?) Example -- TCAS

- TCAS is an airborne system developed by the FAA that
  - operates independently from the ground-based Air Traffic Control (ATC) system
  - designed to increase cockpit awareness of proximate aircraft and to serve as a "last line of defense" for the prevention of mid-air collisions.

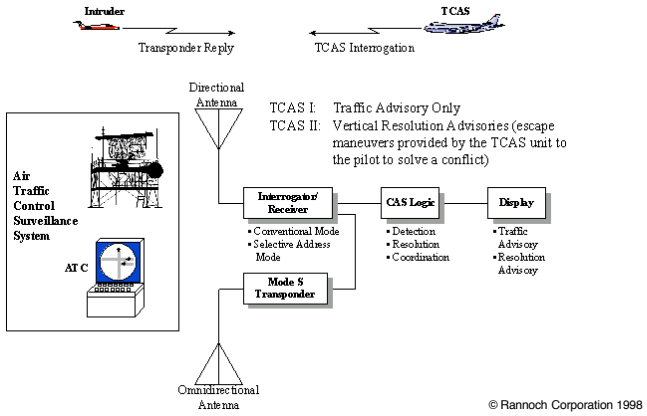
Two levels of TCAS systems

- TCAS I was developed to accommodate the general aviation (GA) community and the regional airlines.
  - issues 'Traffic Advisories' (TAs) to assist pilots in visual acquisition of intruder aircraft.
  - mandated on aircraft with 10 to 30 seats, although TCAS II may be installed instead.
- TCAS II is a more sophisticated system which provides the information of TCAS I **and**
  - analyzes the projected flight path of approaching aircraft and issues 'Resolution Advisories' (RAs) to the pilot to resolve potential mid-air collisions
  - required internationally in aircraft with more than 30 seats or weighing more than 15,000 kg.

TCAS overview



TCAS Overview



Stakeholders in TCAS

- the "innocent bystander" = passengers, people living near airports
- the customer & the end user = commercial and private pilots; air traffic controllers
- the end users' management = airlines, FAA
- the developer = FAA, contractors (system, software), consultants
- the regulators/monitors = NSTB, FAA, Congress

Answers to Stakeholder Questions

- Problem of Providing Satisfactory answers
- Many important questions are vague
  - Is this software user-friendly? How to design appropriate pilot interfaces?

Displays (↓Page 16)

Collision avoidance maneuver advisories and traffic advisories may be displayed to the cockpit crew on one or more dedicated displays, on displays integrated with other instruments such as instantaneous Vertical Speed Indicator (IVSI), or on a CRT flight instrument.

Traffic displays may take several forms: Independent, stand-alone, integrated and time-shared with digital color radar, integrated with the flightcrew's Instantaneous Vertical Speed Indicators (IVSI), or integrated with other displays such as Electronic Horizontal Situation Indicators (EHSI), navigation, or other multi-function displays.

[1.15]
If the traffic display uses a multi-function display that is shared with other services such as ACARS, the traffic display function shall be immediately available for display by a single selection accessible to both pilots.

Answers to Stakeholder Questions

- Is this software user-friendly? How to design appropriate pilot interfaces?

Aural Alerts (→FTA-520, §2.75)

[1.4]
Aural alerts shall be presented by voice announcements only. These announcements are of a prescribed duration.

[1.5]
Aural alerts shall be announced in a high-fidelity distinguishable voice. Automatic volume adjustment for ambient conditions is desired.

[1.6]
RA messages consisting of a single word shall be spoken three times; longer messages shall be spoken twice. If a logic change occurs before the aural alert is complete and a new alert is warranted, the original alert shall be terminated immediately and the new alert started.

[1.7]
Corrective RA messages that indicate that a previously announced corrective RA must be increased in strength or reversed shall be spoken with a sense of urgency.

Visual Alerts (§2.81)

[1.8]
A red visual alert shall be provided in the primary field of view for each pilot for resolution advisories (§2.80).

[1.9]
If a written message is shown on the display, it should flash or otherwise be highlighted. It must be consistent with the aural RA annunciation. Whether

Answers to Stakeholder Questions

- Most important questions have open-ended answers
  - How fast is this system? Different platforms, situations, conditions
- Different stakeholders require different degrees of assurance
  - Affects the degree of thoroughness of (eg.) testing -- Some functions more critical than others
- Different stakeholders have differing degrees of technical sophistication
  - Affects the choice of formalism in which to couch answers
- We'll come back to the issue of representation!

Success of a system

- A system is judged not by properties of the hardware and software, but by the effects of the system in the world
  - you don't care how Caller ID works, just that it works
  - pilots love TCAS (on the whole) because it helps them fly more safely and easily—not because it has great data structures or a fascinating specification

Challenges

1. determine the desired effects (requirements) of the system in the world
   - requirements analysis, requirements discovery, requirements elicitation, requirements, engineering, etc.
   - **extremely** hard to do
2. write this down in an effective way
   - how do you write it down? in what form? does it matter?
     - will help clarify what you think
     - necessary to communicate with customers, other stakeholders
     - forms the basis for a contractual relationship
3. insure that the system satisfies the requirements

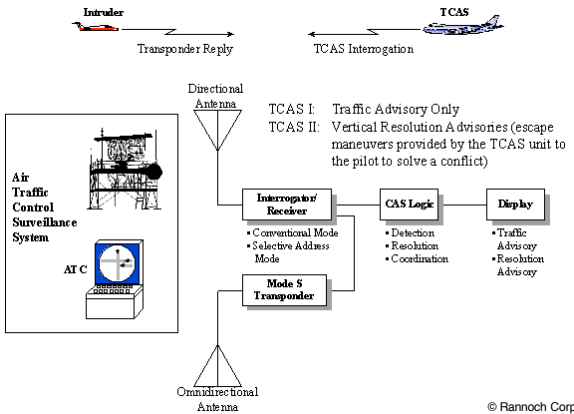
Typical Approach

- Select a (set of [interconnected?]?) representation(s?) (some of which are?) effective in communication with stakeholder constituenc(ies?)
- Derive information/answers expressed in that representation that satisfy stakeholder(s)
- Assure that the information is consistent with other parts of the product (eg. the code!)

How to write it down?

- Back to the TCAS Example
  - FAA needed proof that all collisions will be detected
  - Statement of proof needed
  - Other artifacts: body of proof, code, code structure representations, etc. Must be derived and shown to be consistent with each other

TCAS Overview



Levison "Intentional" Spec

High-Level Functional Requirements

[1.18]
TCAS shall provide collision avoidance protection for any two aircraft closing horizontally at any rate up to 1200 knots and vertically up to 10,000 feet per minute.

Assumption: This requirement is derived from the assumption that commercial aircraft can operate up to 600 knots and 5000 fpm during vertical climb or controlled descent (and therefore two planes can close horizontally up to 1200 knots and vertically up to 10,000 fpm).

[1.19]
TCAS shall handle encounters involving multiple aircraft in areas with large numbers of aircraft within a selected range (without saturation of the operating frequencies).

[1.19.1]
TCAS shall operate in en-route and terminal areas with traffic densities up to 0.3 aircraft per square nautical miles (nmi) (i.e., 24 aircraft within 5 nmi) (↓2.13, ↓Page 202).

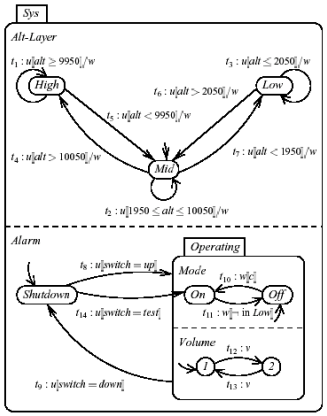
Assumption: Traffic density may increase to this level by 1990, and this will be the maximum density over the next 20 years.

[1.19.2]
TCAS shall operate out to 14 nautical miles (↓Page 188).

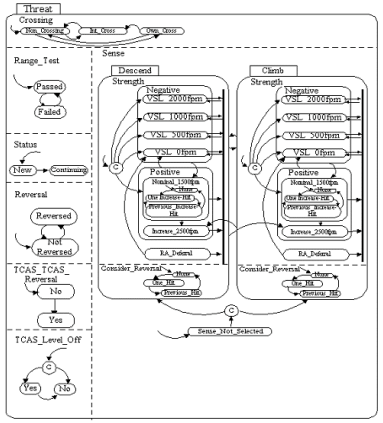
TCAS Truth Tables for Threat_Range_Test

AND	$Other\_Tracked\_Range\_Rate_{i\text{-}540} > 10$ ft/s (RDTHR)	T	T	F	F		
	$Other\_Tracked\_Range_{i\text{-}539} >$ DMOD	F	.	.	.		
	$Modified\_Tau\_Capped_{i\text{-}528} <$ TRTHR	.	.	T	T		
	$Other\_Tracked\_Range_{i\text{-}539} \leq 12.0$ nmi (RMAX)	.	.	T	T		
	$Other\_Tracked\_Range\_Rate_{i\text{-}540}$ * $Other\_Tracked\_Range_{i\text{-}539} >$ H1	F	.	.	.		
	Nuisance_Alarm_Filter	F	.	F	F		
	$Filter\_Status_{s\text{-}300}$ **in state** Dont_Filter_RA	T	T	T	T		
	$Intruder\_Status_{s\text{-}194}$ **in state** Threat	.	T	T	.		
	$Range\_Track\_Firmness_{s\text{-}277}$ **in one of** {3, 4, 5, 6, 7, 8}	.	.	.	T		
	$Range\_Trackers_{s\text{-}287}$ **in state** Not_Initialized	.	T	.	.		

RMSL



TCAS Statechart



SMV program

```
MODULE main
VAR
        u: boolean ;
        v: boolean ;
        switch: {up, down, test};
        alt: 0..20000;
        prev-alt: 0..20000;
        Alt-Layer: {High, Mid, Low};
...
DEFINE
        stable := !(u|v);
        in-Sys := 1;
        in-Alt-Layer := in-Sys;
        in-High := in-Alt-Layer & Alt-Layer = High;
        in-Mid := in-Alt-Layer & Alt-Layer = Mid;
        in-Low := in-Alt-Layer & Alt-Layer = Low;
        in-Alarm := in-Sys;
        in-Shutdown := in-Alarm & Alarm = Shutdown;
...

ASSIGN
init(Alt-Layer) := Mid;
next(Alt-Layer) :=
        case
                t1|t4 : High;
                t2|t5|t6: Mid;
                t3|t7 : Low;
                1 : Alt-Layer;
        esac;
init(Alarm) := Shutdown;
next(Alarm) :=
case
        t8|t14: Operating;
        t9 : Shutdown;
        1 : Alarm;
esac;
...
```

Best laid plans …

- FAA
  - "CAASD personnel have conducted safety studies to evaluate the performance of each successive version of the TCAS logic .."
  - "In a 1997 report on version 7, CAASD's Dr. Michael McLaughlin examined the reduced risk of collision in aircraft equipped with TCAS II versus the risk in aircraft without TCAS … and concluded that
    - "TCAS should reduce NMAC probability by at least 90 to 98 percent," depending on whether one or both aircraft in an encounter are equipped with TCAS."

… go oft astray

- The investigation into the chain of events behind mid-air collision over southern Germany has increasingly focused on the Swiss air traffic control agency Skyguide. Intially Skyguide blamed the Russian crew of one of the two aircraft for ignoring warnings to dive. But since then new important information has come to light: The pilot of the Russian Tu-154 was given conflicting instructions by air traffic control and his onboard computer The Russian pilot was given only 44 seconds warning A warning system at the control centre was switched off for maintenance Only one controller was on duty at the time The centre's radar system does not meet EU standards … BBC