Computers and Risk

Peter B. Ladkin

University of Bielefeld

Middlesex University, 22 April 1997

What is Risk?

This lecture: chances that something will go wrong with projection are *high*

My trip here: chances that something will go wrong with the aircraft flight are *slight*

The *risk* associated with malfunction in the lecture is *slight*

The *risk* associated with malfunction with the aircraft is *high*

So *risk* is not the same as *chance*

I am content to take the chance with the lecture because consequences are not severe.

Were my flight to have an equal chance of disaster I would not be happy flying.

Risk is a function of the *probability* of the event, and the *severity* of the consequences.

$$Risk = f(probability,\ severity, ...)$$

From Leveson, *Safeware*:

An *accident* is "*an undesired and unplanned (but not necessarily unexpected) event that results in (at least) a specified level of loss*"

A *hazard* is "*a state of set of conditions of a system ... that together with other conditions in the environment of the system ... will lead inevitably to an accident (loss event)*"

The *danger* of a hazard is the "*likelihood of the hazard leading to an accident*"

The *latency* of a hazard is the "*hazard exposure or duration*"

The *severity* of a hazard is "*the worst possible accident that could result from the hazard given the environment in its most unfavorable state*"

$$\textit{Hazard level} = h(\textit{probability, severity})$$

and

$$\textit{Risk} = g(\textit{hazard level,danger,exposure})$$

How may we measure these? With difficulty.

Before we do, a couple of views on
risk characterisation:

"*..the process of estimating the incidence of a health effect under the various conditions of human exposure described in exposure assessment. It is performed by combining the exposure and dose-reponse assessments.*"
from *Risk Assessment in the Federal Government: Managing the Process*, National Academy Press, 1983.

"*...a synthesis and summary of information about a potentially hazardous situation that addresses the needs and interests of decision makers and of interested and affected parties. [It] is a prelude to decision-making and depends on an iterative analytic-deliberative process.*"
from *Understanding Risk: Informing Decisions in a Democratic Society*, National Academy Press, 1996.

Everything clear?

Probability of an event.

Consider safety-critical systems: an aircraft

Acceptable probability for reliability:
1 failure in 10^{-9} 'cycles' Reliability must be is very, very high!

Littlewood and Strigini have argued that at this level, the *a priori* probability cannot be raised at all by testing - life is too short!

Conclusion:
one must establish this figure *a priori*

Note: reliability and safety are not the same!

Computer epidemiology:

Assumption: bugs reveal themselves in normal operation with specific frequency. Operate the device, catch the frequent bugs; what's left are the infrequent bugs. We can live with those. (You rather than me.)

And remember, not just probability but *severity*.

How to measure *Severity*?

Sociological questions.

One particularly discomforting answer:
the cost of a life - what would have to be spent to save one more life, *mutatis mutandis*

But not all lives are the same. Harm to my spouse is more significant to me than harm to someone else's spouse in Bosnia.

To evaluate severity, people have

- Preferences
- Understanding

The ultimate

Death

sudden — lingering

painful — painless

uncomfortable — comfortable

Preference:
A masochist might prefer *painful*
I prefer *painless*

Understanding:

Many people believe that an aircraft falls out of the sky when the engines quit.

Hazard: engines quit.
Danger: highest? (in fact, moderate)
Severity: very high? (in fact, like an auto accident)

China Air, Pan Am 103

A computer is an *artifact*. What is an artifact?

Consider a screwdriver.

Physical object; but not sufficient.

Anything of that rough form would not necessarily be a screwdriver

It has *function*, which has to do with *purpose*

Bestowing purpose is an *intentional act*, performed by an *agent*.

It is a type of *speech act* (Austin, Searle).

What distinguishes a computer or an aircraft from a screwdriver?

The computer and aircraft are designed to exhibit *variable behavior*.

A screwdriver is designed to be used in someone else's behavior, but its *state* remains more or less the same. You can look at a screwdriver and tell if it will fulfil its purpose.

A computer or aircraft is designed to exhibit behavior *sui generis*. You cannot so easily tell by looking. You have to observe over time.

What you can describe by looking is called *state*.

Look, observe a state. Look again later, observe another state. This second state may be different, but much may be the same. And again, and again.

Such a unending sequence of states is called a *behavior*. There may be more than one (technical) behavior describing a behavior, depending on when we choose to look.

An artifact has a purpose.
It is supposed to do certain things, in order, maybe depending on if other things have been done, or if the state is appropriate.

We can describe these individual actions also by purpose (a sort of decomposition).

What are the conditions for starting?
How do we know when it's succeeded?

$$\text{Action} \stackrel{\triangle}{\Leftrightarrow} < \textit{start–condition}, \textit{finish–condition} >$$

This is how computer scientists started.
But it's too inflexible. We need to describe a *relation* between starting and finishing values.

Consider a message-forwarding system

Two actions:

push(some-message) and *pop*

The system state shows accepted messages in sequence waiting to be conveyed.

The figure shows the state of the message system before each operation and after.

$$\begin{array}{ccc} \text{buffer} & & \text{buffer}' \\ < \text{b,x,y,d} > & \xrightarrow{\text{pop}} & < \text{x,y,d} > \end{array}$$

$$< \text{b,x,y,d} > \xrightarrow{\text{push(a)}} < \text{b,x,y,d,a} >$$

Here is a concrete version of the system

There are permanent slots for messages which must be shuffled along from slot to slot, put on the back, and popped off the front.

$\perp$ is the symbol for ‘*nothing here*’

Buffer		Buffer′
$\perp$ b x y $\perp$ d $\perp$	$\xrightarrow{\text{push(a)}}$	$\perp$ b x y $\perp$ d a
$\perp$ b x y $\perp$ d a	$\xrightarrow{\text{move(6)}}$	$\perp$ b x y d $\perp$ a
$\perp$ b x y d $\perp$ a	$\xrightarrow{\text{move(2)}}$	b $\perp$ x y d $\perp$ a
b $\perp$ x y d $\perp$ a	$\xrightarrow{\text{pop}}$	$\perp$ $\perp$ x y d $\perp$ a

The concrete system has much more detail than the abstract system. So it should be.

Here's how the concrete system is supposed to correspond to the 'abstract' system in terms of state:

ConcreteBuffer $\qquad\qquad$ AbstractBuffer

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & \perp & d & \perp \\ \hline \end{array} \longrightarrow < b,x,y,d >$$

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & \perp & d & a \\ \hline \end{array} \longrightarrow < b,x,y,d,a >$$

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & d & \perp & a \\ \hline \end{array} \longrightarrow < b,x,y,d,a >$$

$$\begin{array}{|c|c|c|c|c|c|c|}\hline b & \perp & x & y & d & \perp & a \\ \hline \end{array} \longrightarrow < b,x,y,d,a >$$

And here's how the actions correspond:

$$\begin{array}{ccc} \text{Buffer} & & \text{Buffer}' \end{array}$$

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & \perp & d & \perp \\ \hline \end{array} \xrightarrow{\text{push(a)}} \begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & \perp & d & a \\ \hline \end{array}$$

$$< \text{b,x,y,d} > \xrightarrow{\text{push(a)}} < \text{b,x,y,d,a} >$$

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & \perp & d & a \\ \hline \end{array} \xrightarrow{\text{move(6)}} \begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & d & \perp & a \\ \hline \end{array}$$

$$< \text{b,x,y,d,a} > \xrightarrow{\texttt{no-op}} < \text{b,x,y,d,a} >$$

$$\begin{array}{|c|c|c|c|c|c|c|}\hline \perp & b & x & y & d & \perp & a \\ \hline \end{array} \xrightarrow{\text{move(2)}} \begin{array}{|c|c|c|c|c|c|c|}\hline b & \perp & x & y & d & \perp & a \\ \hline \end{array}$$

$$< \text{b,x,y,d,a} > \xrightarrow{\texttt{no-op}} < \text{b,x,y,d,a} >$$

$$\begin{array}{|c|c|c|c|c|c|c|}\hline b & \perp & x & y & d & \perp & a \\ \hline \end{array} \xrightarrow{\text{pop}} \begin{array}{|c|c|c|c|c|c|c|}\hline \perp & \perp & x & y & d & \perp & a \\ \hline \end{array}$$

$$< \text{b,x,y,d,a} > \xrightarrow{\text{pop}} < \text{x,y,d,a} >$$

How does this fit together?

These are both *state machines*:

- state changes happen only via
  a fixed set of specified operations;

- they deal with a fixed finite
  set of state variables.

The concrete system *simulates* the 'abstract'

- they start in 'equivalent' states;

- every action of the concrete system
  corresponds either to an action
  or to a non-action of the abstract system;

- when the concrete system is sufficiently
  'live', then the abstract buffer is also.

We must choose a *system boundary*:
a partition between what we change to achieve the purpose, and what is left alone.

$$\text{The world} = \textit{System} \oplus \textit{Environment}\ ?$$

$$\textit{System} \overset{\triangle}{\Leftrightarrow} \text{Changes in state variables belonging to } \textit{System}$$

$$\textit{Environment} \overset{\triangle}{\Leftrightarrow} \text{Changes in state variables belonging to } \textit{Environment} \text{ (mostly irrelevant)}$$

But what about the mixture?

$$
\begin{array}{lll}
\text{buffer} & & \text{buffer}' \\[1em]
\{\} \quad < \text{b,x,y,d} > \{e,f\} & \xrightarrow{\text{pop}} & \{b\} \quad < \text{x,y,d} > \quad \{e,f\} \\[1em]
\{b\} \quad < \text{x,y,d} > \quad \{e,f\} & \xrightarrow{\text{push}} & \{\} \quad < \text{x,y,d,f} > \{e\} \\[1em]
\{\} \quad < \text{x,y,d,f} > \{e\} & \xrightarrow{\text{pop}} & \{x\} \quad < \text{y,d,f} > \quad \{e\} \\[1em]
\{x\} \quad < \text{y,d,f} > \quad \{e\} & \xrightarrow{\text{pop}} & \{y\} \quad < \text{d,f} > \quad \{e\}
\end{array}
$$

Once it has received the *pop*'d message, the environment does what it likes with it (in this case, often loses it).

But what happens if they don't work together?

$$\text{buffer} \qquad\qquad\qquad\qquad \text{buffer}'$$

$$\{\} \quad < \text{b,x,y,d} > \{e,f\} \xrightarrow{\text{pop}} \{a\} \; < \text{x,y,d} > \quad \{e,f\}$$

$$\{a\} \; < \text{x,y,d} > \quad \{e,f\} \xrightarrow{\text{push}} \{\} \quad < \text{x,y,d,q} > \{e\}$$

$$\{\} \quad < \text{x,y,d,q} > \{e\} \xrightarrow{\text{pop}} \{p\} \; < \text{y,d,q} > \quad \{e\}$$

$$\{p\} \; < \text{y,d,q} > \quad \{e\} \xrightarrow{\text{pop}} \{p,t\} < \text{d,q} > \quad \{e\}$$

The environment and the message system just aren't working together!

$$\mathit{Interface} \stackrel{\triangle}{\Leftrightarrow} \text{Changes involving state variables of both } \mathit{Sys} \text{ and } \mathit{Env}$$

$$\text{The world} = \mathit{System} \oplus \mathit{Environment} \oplus \mathit{Interface}$$

This can be iterated.

$$\mathit{Subsystem}_1 = \mathit{Subsystem}_2 \oplus \mathit{Subsystem}_3 \oplus \mathit{Interface}_{\{2,3\}}$$

$$\mathit{Subsystem}_6 = \mathit{Subsystem}_1 \oplus \mathit{Human}_4 \oplus \mathit{Interface}_{\{1,4\}}$$

$$\mathit{System} = \mathit{Subsystem}_6 \oplus \mathit{Local\text{-}`Weather'} \oplus \mathit{Interface}_{\{6,Weather\}}$$

$$\text{The world} = \mathit{System} \oplus \mathit{Environment} \oplus \mathit{Interface}$$

Systems have *parts*; parts are subdivisions of behavior, which may be subdivisions of state variables

Subdivisions of state variables are what we normally think of as 'parts'

But don't forget the behavior divisions which cannot be obtained by restricting to these state-variable partitions.

That's the *interface*.

If there are 4 subsystems, there are $2^4 = 16$ pairwise interfaces, and $choose(3, 4) = 4$ three-wise interfaces.

If there are 8 subsystems, there are $2^8 = 256$ pairwise interfaces, and $choose(3, 8) = 56$ three-wise interfaces.

That's why engineers earn a lot of money!

Physical engineering: state is predictable

x-value in any state t:

$$\int f(t,y)dy\ dt$$

Discrete systems: say a system with one variable x, taking values $0, 1$.

Start in state { x = 0 }

Next state could be { x = 0 } or { x = 1 }.

So could the next one.

After 64 actions, the behavior could have been any one of 2^{64} possible behaviors. That's more than the number of elementary particles in the solar system (so I understand).

Suppose a discretely-controlled system ‘goes wrong’. Say, it gets in a state which is not part of the planned behavior. The physical computer does not break. It continues to operate ‘normally’, but in ‘unforeseen ways’. (This is what is meant by engineers who say that software ‘doesn’t fail’.)

Its ‘actions’ don’t relate any more to the purpose. But it still continues to act. Actions have ‘random’ effects, because they no longer fit within the descriptive framework provided by the purpose.

What causes people to worry particularly about computers going wrong?

There are three main ways computer problems distinguish themselves from physical failures.

- they're always going wrong, despite mostly fanatical devotion of programmers (Win95, Word, "Plug and pray", Ariane 5);

- when they do wrong, analysis is at best extremely difficult (Space Shuttle incident);

- they don't go wrong in predictable ways (see forthcoming Y2K problem)

These can mostly be explained by the *combinatorial complexity* feature: lots of behaviors, lots of interfaces.

Will we get better at monitoring and controlling? Yes. But it requires extensive automated reasoning capabilities that we do not yet have.

I can prove that my concrete ‘design’ implements my ‘abstract’ requirement.

They’re not ‘concrete’ and ‘abstract’, though. They’re both described in the same way.

By further detailing and decomposition
I can build a hierarchy of detailed machines the bottom of which corresponds to the physical operation of an (uncomplicated) chip and the top of which corresponds to the purpose.

The bottom chip is pure hardware: it's like a light bulb: if it doesn't function, we throw it away (presuming we can diagnose it!)

The top is the *statement of purpose*.
But how do we know if we got it right?
(Lufthansa Warsaw, Ariane)

That's the art of *requirements specification*.
'Urban legend': a JPL study of faults in safety-critical systems found 97+% in mismatch between stated and intended purpose!

Reliability: fulfilling the purpose

Safety: ensuring nothing bad happens

If an aircraft has engine problems and cannot be used in revenue service, then it is *not reliable* but it is *safe*

Modern trend is to automate functions previously performed mechanically, hydraulically, or by humans.

Gains:
simplicity, component reliability (:-),
weight savings, on-the-fly redesign (:-?,
optimal manœuvring, functionality (:-?

Losses:
Simplicity, component reliability (:-(

When everything works, piloting is easier.
When something goes wrong, our old friend
combinatorial explosion
(milder than the real thing,
but can be just as deadly)

Numbers of Aircraft
(from Flight International, 4-10 Dec 1996)

A320 (intro March 1988):	Ordered	1,110
(including A319/A321)	Delivered	613
A330	Ordered	160
	Delivered	47
A340	Ordered	178
	Delivered	87
A300	Ordered	485
	Delivered	460
A310	Ordered	271
	Delivered	252

B727	Delivered	1,831
B737-100/200	Delivered	1,144
B737-300/400/500	Ordered	1,922
	Delivered	1,681
B737-600/700/800	Ordered	426
B747-100/200/SP	Delivered	724
B747-400	Ordered	528
	Delivered	367
B757	Ordered	854
	Delivered	732
B767	Ordered	724
	Delivered	630
B777	Ordered	281
	Delivered	40
BAC111	Delivered	232
BA146	Delivered	219
Avro RJ70/85/100/115	Ordered	105
	Delivered	72

Some Examples:

Common Mode:	Aeroperu.
HCI:	Cali, Toulouse, Birgenair?, Nagoya? Aeroperu?
Requirements:	Warsaw.
Systems:	A340 Heathrow
Pilot Error:	Toulouse, Birgenair, Cali, Warsaw.