



# Software Quality: Exception Handling

# The \$440 Million Software Error at Knight Capital



What happened to Knight on the morning of August 1, 2012, is every CEO's nightmare: A simple human error, easily spotted with hindsight but nearly impossible to predict in advance, threatened to end the firm.

At Knight, some new trading software contained a flaw that became apparent only after the software was activated when the New York Stock Exchange (NYSE) opened that day. The errant software sent Knight on a buying spree, snapping up 150 different stocks at a total cost of around \$7 billion, all in the first hour of trading.

Error: "A human action that produces an incorrect result."

Defect: "Deviation or irregularity from the specifications."

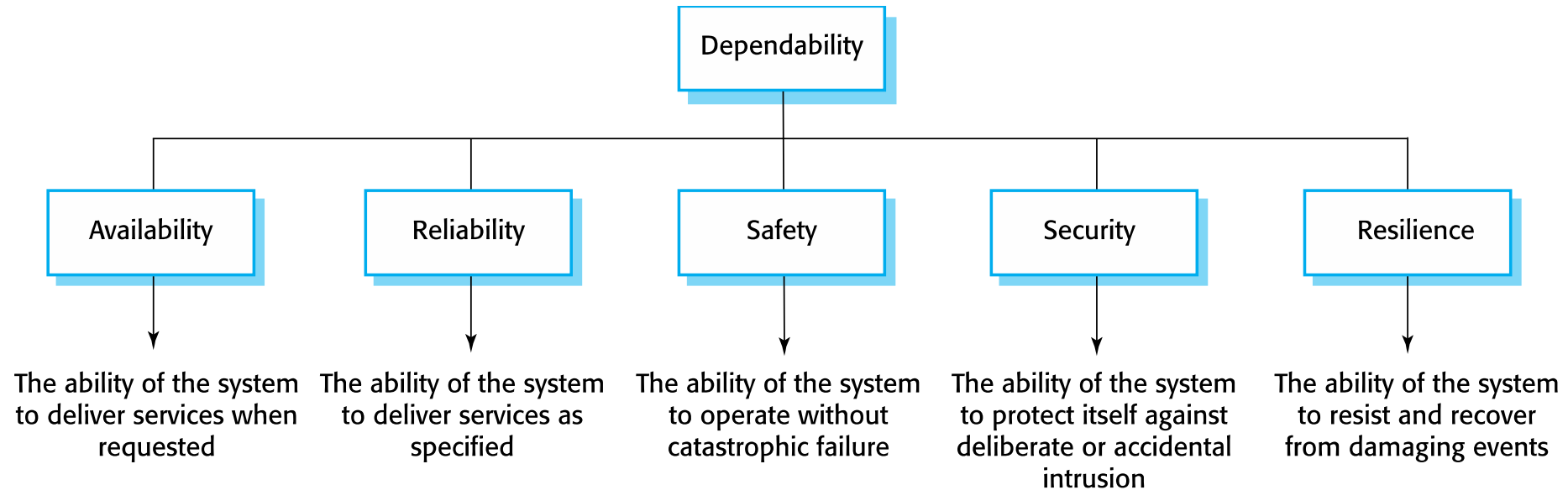
Failure: "An event in which a component or system does not perform a required function within specified limits."

Bug: "Flaws that impact on software functionality and performance."

- For many computer-based systems, the most important system property is the dependability of the system.
- The dependability of a system reflects the user's degree of trust in that system. It reflects the extent of the user's confidence that it will operate as users expect and that it will not 'fail' in normal use.
- System failures may have widespread effects with large numbers of people affected by the failure.

- Systems that are not dependable and are unreliable, unsafe or insecure may be rejected by their users.
- The costs of system failure may be very high if the failure leads to economic losses or physical damage.
- Undependable systems may cause information loss with a high consequent recovery cost.

- Hardware failure
  - Hardware fails because of design and manufacturing errors or because components have reached the end of their natural life.
- Software failure
  - Software fails due to errors in its specification, design or implementation.
- Operational failure
  - Human operators make mistakes. Now perhaps the largest single cause of system failures in socio-technical systems.





# ESA Ariane 5 Flight V88, 1996



```
numeros = [42]
```


```
numeros = [42]
```

```
assert numeros[0] != 42, "Esperamos um número diferente de 42"
```

```
assert 42 in numeros, "Esperamos o elemento 42 na lista"
```

```
assert isinstance(numeros[0], int), "Esperamos um número inteiro"
```

```
assert type(numeros[0]) == int, "Esperamos um número inteiro"
```

- Verificação de pré e pós condições
- Verificações de invariantes
- Verificações de sanidade (Sanity Check)
- Todos os casos acima  Depuração

- Assertivas não devem ser utilizadas para processamento ou validação de dados
- Assertivas não devem ser utilizadas para tratamento de erro
- Assertivas não devem ser utilizadas em produção



# Dúvidas?