

---

# **Patterns for Technical Software Design**

## **A provisional catalogue and two case studies**

**Michael J. Pont**

Engineering Department, Leicester University

Leicester University, Department of Engineering,  
**Technical Report 97-25, December 1997.**

**Based on a research seminar presented to:**

Computer Science Research Group,  
Department of Mathematics and Computer Science,  
Leicester University.

5<sup>th</sup> December, 1997

---

## OVERVIEW

- This talk will consider whether the success rate of technical software projects could be enhanced through the use of appropriate 'design patterns'.
- The origin of design patterns, and how they may be used to support software reuse, will be discussed.
- A provisional 'catalogue' of patterns suitable for use technical software projects will be presented
- The use of this catalogue illustrated through two case studies

## NOTE:

- The talk will describe 'work in progress'
- Comments / suggestions on the ideas presented would be welcomed
- A copy of these OHPs is available on the WWW (<http://www.le.ac.uk/engineering/mjp9>)

---

## **OVERVIEW**

**1. What is Technical Software?**

**2. Software Design**

**3. Example: Cruise Control**

**4. The Root of the Problem - and a possible solution**

**5. The Origins of (software) Design Patterns**

**6. Example: 'Window place'**

**7. Related ideas**

**8. Example: 'Observer'**

**9. Other Design Patterns**

**10. A provisional catalogue of patterns for technical software design**

**11. Case study: Medical Decision Support (EMG)**

**12. Case study: Engine Fault Diagnosis**

**13. Conclusions**

---

## ~~ALL MY OWN WORK~~

The case studies outlined in this talk were carried out in conjunction with:

### Medical Decision Support (EMG)

Emanuela Moreale, S.Q. Wang

Barrie Jones, Paul Mocroft<sup>1</sup>, John Fothergill, Fernando Schindwein

### Engine Fault Diagnosis

Lee Walton

John Fothergill, Barrie Jones

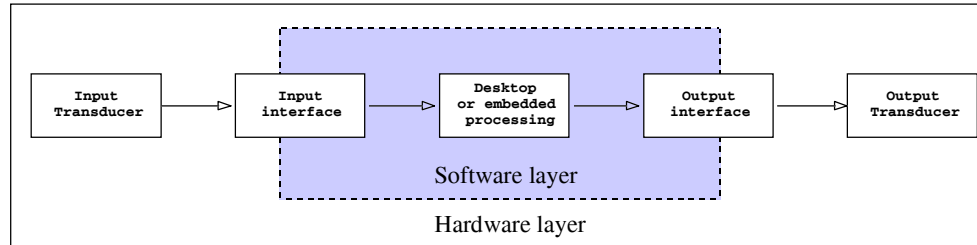
All Department of Engineering, Leicester University, except where noted:

<sup>1</sup>*The Queen Elizabeth Hospital, Birmingham.*

---

## WHAT IS TECHNICAL SOFTWARE?

The primary focus of this talk is on sampled-data applications, taking the following general form:



For example:

- Embedded, real-time, ECG monitoring
- On-line engine fault diagnosis
- Speech recognition
- Cruise control
- Face recognition
- Stock-market price prediction.
- Digital radio
- Control of a steel press
- ...

---

## SOFTWARE DESIGN

- Software design methodologies were created in the 1970s to support the development of information systems in a commercial context.
- The original techniques have now been modified and extended to allow their use for the development of technical systems.

---

## EXAMPLE: **CRUISE CONTROL**

- A cruise-control system is required to take over the task of maintaining the vehicle at a constant speed even while negotiating a varying terrain, involving, for example, hills or corners in the road.
- Subject to certain conditions (typically that the vehicle is in top gear and exceeding a preset minimum speed), the cruise control can be engaged by the driver via a push switch on the dashboard, and disengaged by touching the brake pedal.

# EXAMPLE: CRUISE CONTROL

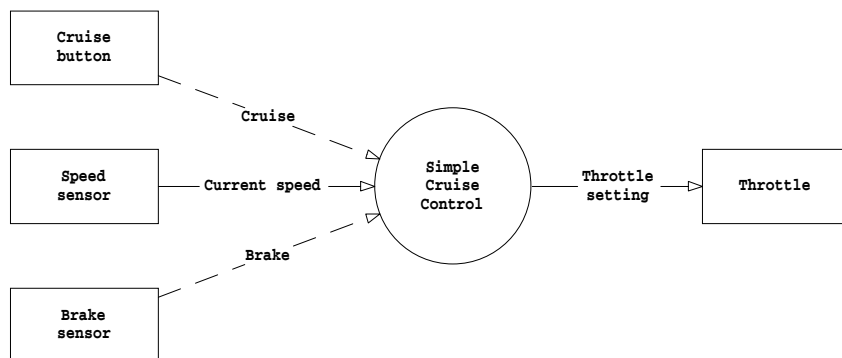


Figure 1: The context diagram for the simple cruise control system.

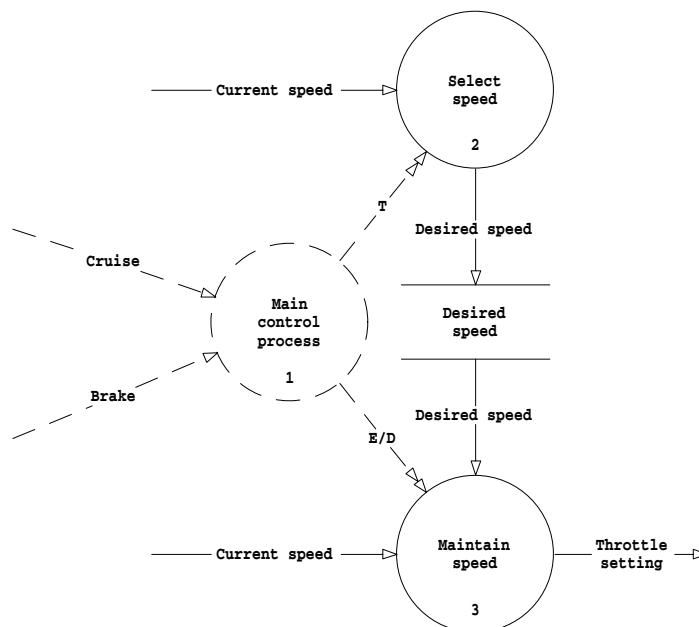


Figure 2: The Level 1 dataflow diagram for the cruise control system.

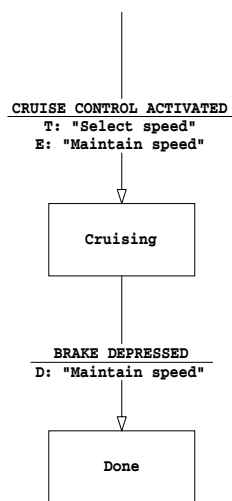


Figure 3: The state-transition diagram associated with the control process in Figure 2.

---

## EXAMPLE: CRUISE CONTROL

### PSpec - Maintain Speed

Set:

$$V_{Th} = \begin{cases} 0; & (S_D - S_A) > 2 \\ 2(S_D - S_A + 2); & -2 \leq (S_D - S_A) \leq 2 \\ 8; & -2 > (S_D - S_A) \end{cases}$$

Subject to:

$$\frac{dV_{Th}}{dt} \leq 0.8V / \text{sec}$$

Where:

$V_{Th}$  = Throttle setting

$S_D$  = Desired speed

$S_A$  = Current Speed

The throttle setting is assumed (here) to be proportional to an output voltage: a 0V output closes the throttle, and an 8V output sets it fully open.

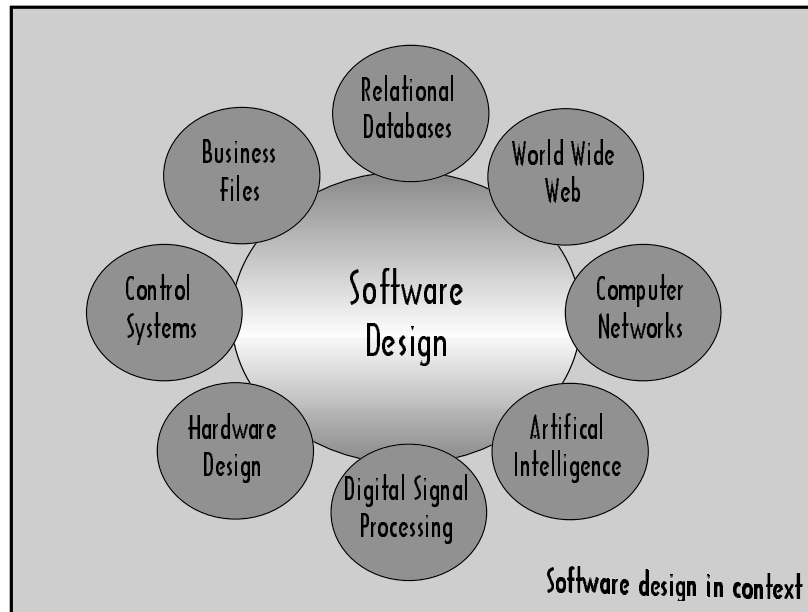
The intention is to vary the throttle setting when the speed varies by more than 2mph above or below the desired speed. The rate of throttle opening is restricted to 0.8V / second.

[Adapted from Hatley and Pirbhai (1987), p.291.]

**What is described above is essentially a proportional (only) control system: such simple control systems are prone to a number of problems, including significant steady-state errors...**

---

## THE ROOT OF THE PROBLEM - AND A POSSIBLE SOLUTION



- The problem of placing software design 'in context' arises during the development of a wide range of software systems

---

## THE ROOT OF THE PROBLEM - AND A POSSIBLE SOLUTION

**Unlike information systems, the successful design of technical systems requires not just knowledge of software engineering, but also contributions from related technical and engineering fields, including instrumentation, (digital) signal processing, control theory, artificial intelligence, statistics, ...**

### **This is a problem because:**

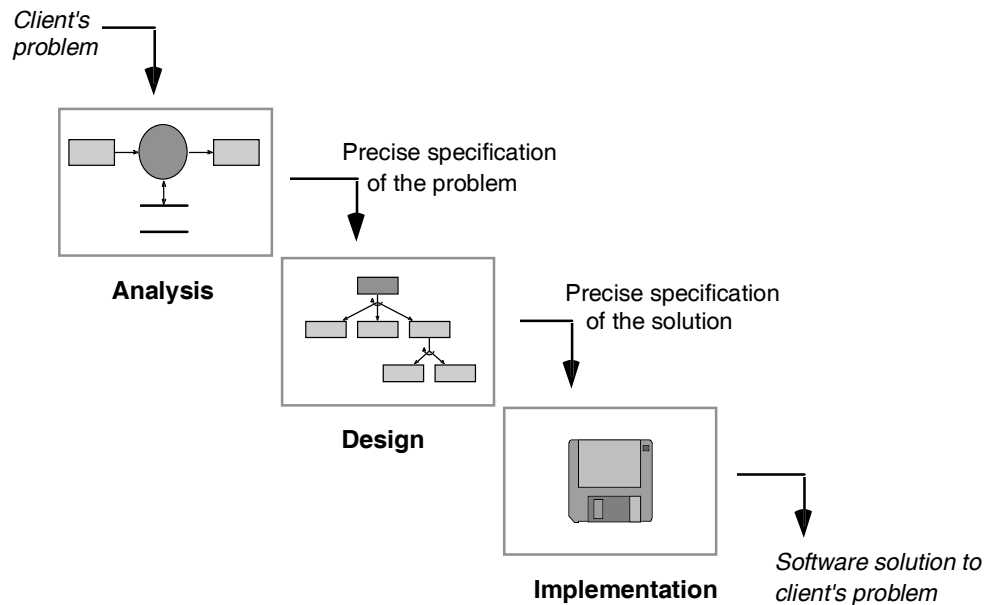
- Software engineering is a comparatively young discipline, one consequence being that design handbooks, common in other engineering disciplines such as electronic engineering are not yet available.
- The majority of software engineers who are currently practising (and have been formally trained) were taught in computer science departments, and often have little knowledge of topics outside the traditional boundaries of computer science or software design.
- The increasingly ubiquitous nature of software means that no individual can be expected to be knowledgeable in all areas relevant to / related to software design.

**Also...**

---

## THE ROOT OF THE PROBLEM - AND A POSSIBLE SOLUTION

- Like software design methodologies, software process models were originally developed to support the development of information systems



- For example, this 'waterfall' process model reflects the 1970s situation, when information systems were being developed to replace paper-based systems.
- Now, for most systems, useful code and/or useful designs already exists in the company and/or the public domain
- We need to stop reinventing the wheel
- We need to encourage 'software reuse'

**Can 'design patterns' contribute to a solution?**

---

## THE ORIGINS OF (SOFTWARE) DESIGN PATTERNS

- Patterns for software design have their roots in links between software design and contemporary architecture.
- Many people now argue that ‘software architect’ is a more appropriate job title than ‘software engineer’ ...
  
- The architect Christopher Alexander (e.g. Alexander et al., 1977) first described what he called ‘a pattern language’ relating various architectural problems (in buildings) to good design solutions.
- This basic concept was adopted by Ward Cunningham and Kent Beck who used some of Alexander’s techniques as the basis for a small ‘pattern language’ intended to provide guidance to novice Smalltalk programmers (Cunningham and Beck, 1987).
- This work was subsequently built on by Erich Gamma and colleagues who, in 1995, published an influential book on general-purpose object-oriented design patterns (Gamma *et al.*, 1995).

---

## EXAMPLE: 'WINDOW PLACE'

### Problem

- For a hotel / office / large house you need to design a 'living room' in which people will congregate to sit and talk, drink tea, read newspapers, etc...

### Solution

- Ideally, in the 'living room', create a 'window place'.

### Notes / Comments

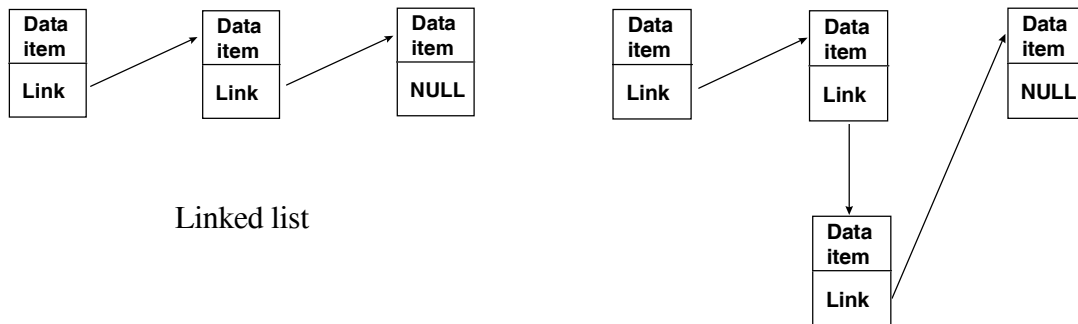
- During the day, people generally dislike spending time in rooms without windows...
- When a room has windows, then people will be drawn to them: if the seats are adjacent to the windows in your living room (preferably so that people can see out from their seats), then people will feel comfortable in this 'window place'.
- If the windows are on one side of the room and the seats on the other then people will feel uncomfortable

[Adapted from Alexander *et al.*, 1977]

---

## RELATED IDEAS

- Many computer scientists / software engineers are taught about data structures (linked lists, queues, trees, etc).



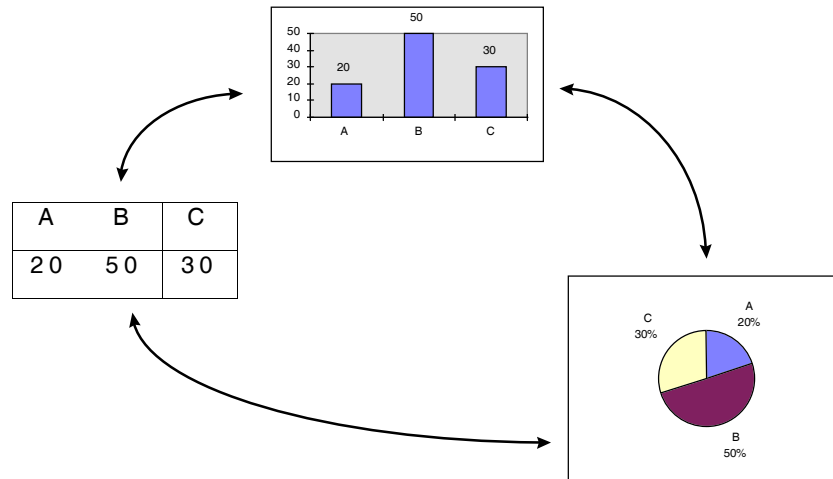
- Design patterns take these (useful) ideas one step further
- Design patterns are intended to link 'problems' to 'solutions'...

---

## EXAMPLE: 'OBSERVER'

### Problem 1

- In a spreadsheet, the same information may be displayed (for example) as table of data, as a bar chart and as a pie chart. Changes to any of these displays need to be reflected in the others:

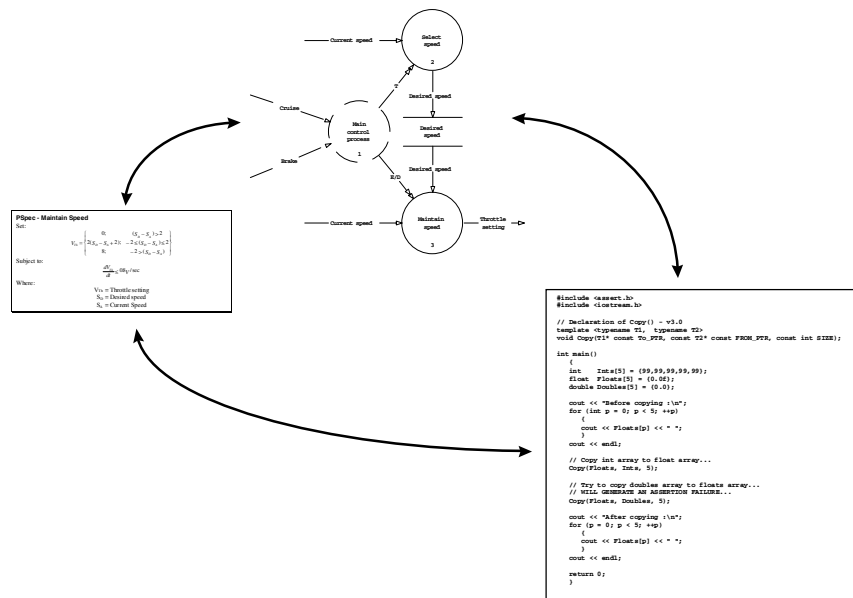


[Adapted from Gamma *et al.*, 1995]

# EXAMPLE: 'OBSERVER'

## Problem 2

- In a CASE tool, a system can be represented (for example) using a combination of dataflow diagram(s), process specifications and code. Changes to any one of these displays may require changes to the others.





---

## OTHER DESIGN PATTERNS

### See:

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley, Reading, MA.

Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. (1996) “Pattern-Oriented Software Architecture: A System of Patterns”, John Wiley and Sons.

### Example: Abstract Factory

Used, for example, to create user interface toolkits that support multiple ‘look and feel’ standards (like Windows 3.1, Windows 95, Mac, Motif, Presentation Manager, ...)

[Gamma *et al.*, 1995]

### Example: Master-Slave

Used to support parallel computation. A master component distributes work to identical slave components and computes a final solution from the results these slaves return.

[Buschmann *et al.*, 1996]

---

## A PROVISIONAL CATALOGUE OF PATTERNS FOR TECHNICAL SOFTWARE DESIGN

**There are nine patterns in this provisional catalogue:**

- ‘Input’
- ‘Output’
- ‘Transform’
- ‘Filter’
- ‘Extract’
- ‘Compare’
- ‘Model’
- ‘Classify’
- ‘Control’

**Note:**

- By definition, patterns are **not** new...
- The patterns described here arose from my experience on a wide range of software projects over the last decade
- Typically several of these patterns (and other design patterns) will be used on a single project: this will be demonstrated in the case studies at the end of this talk

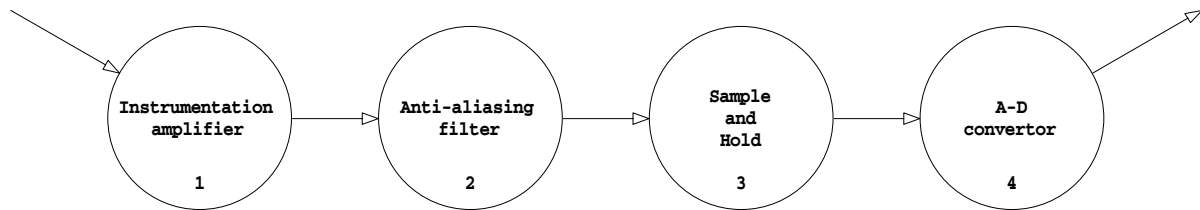
---

# 'INPUT'

## Problem

In most sampled-data systems, it is necessary to convert a continuous-time signal into a discrete-time equivalent.

## Solution



## Notes / Comments

- 'Input' straddles the software / hardware divide
- Key considerations are the selection of (1) an appropriate sampling rate, and (2) an appropriate anti-aliasing filter.

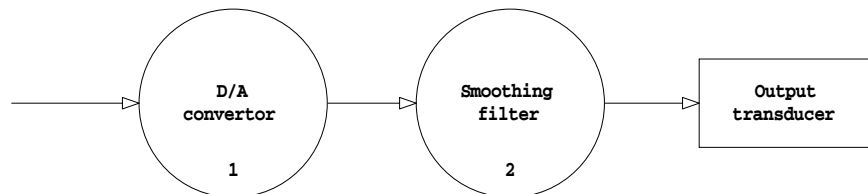
---

# 'OUTPUT'

## Problem

In many sampled-data systems, it is necessary to convert a discrete-time signal into a continuous-time equivalent.

## Solution



## Notes / Comments

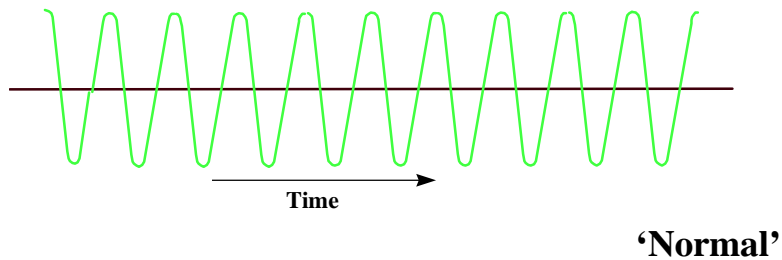
- 'Output' straddles the software / hardware divide
- A key consideration is the selection of an appropriate smoothing filter.

---

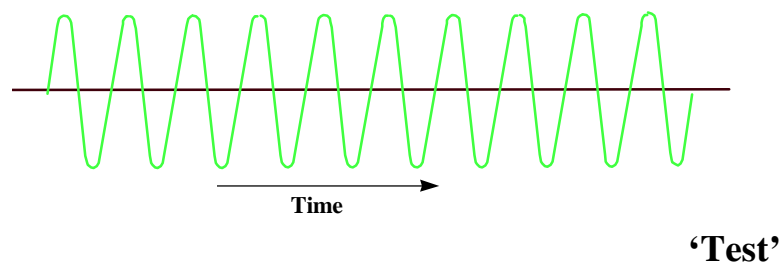
## 'TRANSFORM'

### Example

As a result of a machine monitoring operation, we have the following vibration signature representing the 'normal' operation of a piece of textile machinery:

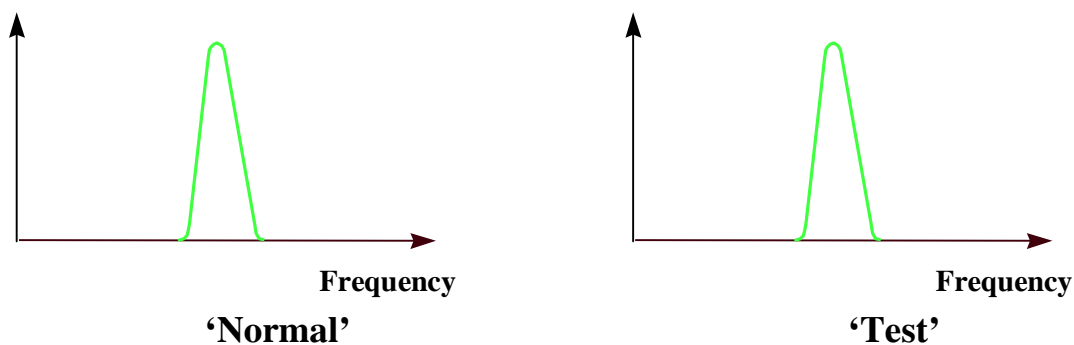


We subsequently obtain the following signal from a test of the machine:



A direct comparison of the two signals (see 'Compare') on a sample-by-sample basis will suggest that the test and normal signals differ significantly and that the machine has a fault condition.

However, transforming these signals into the frequency domain will eliminate the (phase) differences, and reveal that both signals are very similar:

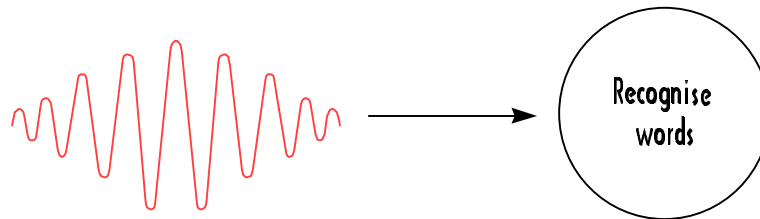


---

## 'TRANSFORM'

### Example

Speech recognition is a challenging problem. Generally, from the speech waveform alone (for example, the time-varying voltage waveform from a microphone), we aim to recognise the words spoken:



The human ear performs a time-domain to frequency-domain conversion of the speech waveform at an early stage of human auditory processing.

Similarly, almost all successful speech recognition systems involve transforming the speech waveform (to give a frequency-domain representation) prior to carrying out the recognition task.

---

## **‘TRANSFORM’**

### **Problem**

In many situations, it is helpful to convert signals from the time-domain into the frequency domain (and vice versa).

### **Solution**

- Many solutions are based on the (discrete) fast Fourier transform (FFT).
- For very short, or rapidly-changing signals, the wavelet transform offers some advantages.

### **Notes / Comments**

### **Related patterns**

Extract, Filter, Compare

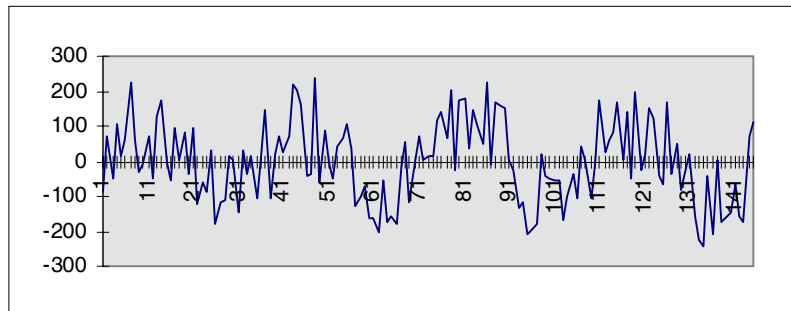
---

# 'FILTER'

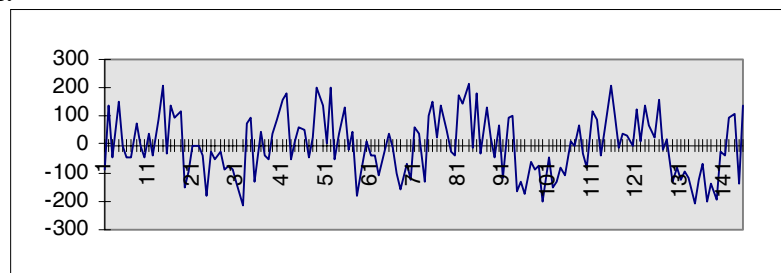
## Example

A recording of vibration signals from a piece of industrial machinery is contaminated by vibrations (mainly through the floor) from neighbouring pieces of equipment.

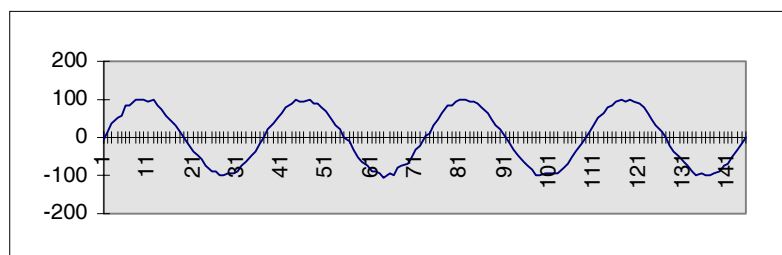
First signal:



Second signal:



Because the noise and signal are not correlated, we can 'filter' out the noise by averaging the results of 1000 signals:



---

## **'FILTER'**

### **Problem**

Filter is used to remove unwanted features from signal S.

### **Solution**

- In most circumstances, we are interested in digital filters
- Two broad categories are IIR and FIR
- The 'filter' operation can also - as in the first example - be carried out by signal averaging.

### **Notes / Comments**

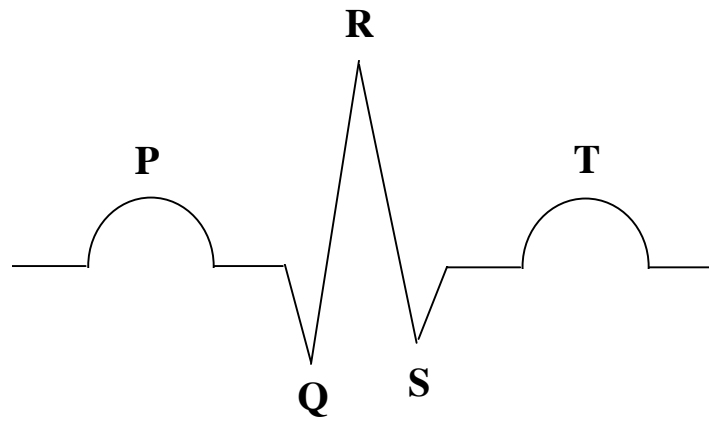
### **Related patterns**

---

## **'EXTRACT'**

### **Example**

Analysis of the electrocardiogram (ECG) is usually carried out by identifying features in the waveform:



---

## **‘EXTRACT’**

### **Problem**

In many condition monitoring / fault diagnosis systems, we wish to ‘extract’ from the signal significant events / features which we are interested in analysing further.

### **Solution**

No general solution

### **Notes / Comments**

- Many problems are split into feature extraction followed by classification...
- Selection of appropriate features (and feature extraction techniques) is usually based on knowledge of the problem domain.

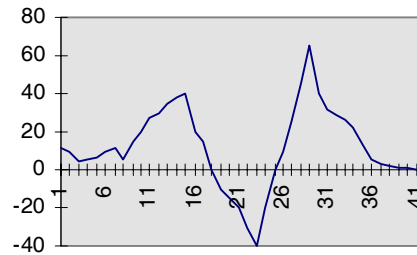
### **Related patterns**

---

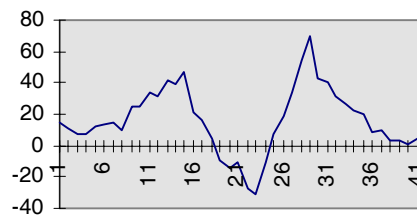
# 'COMPARE'

## Example

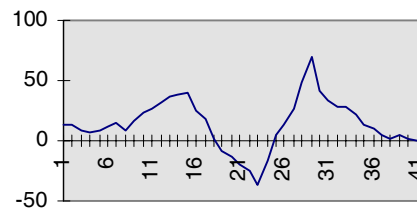
In a medical test, after signal averaging, we obtain the following signal:



A month later, from the same patient, we obtain this signal:

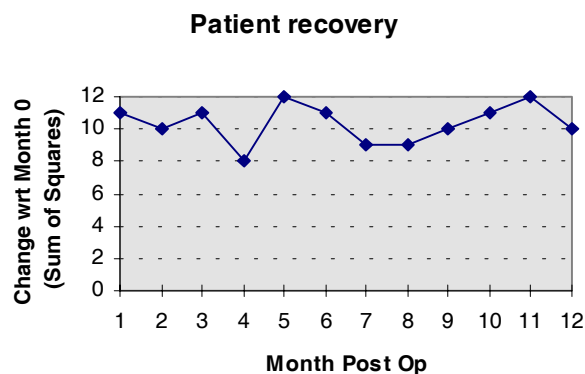


A further month later, we obtain this signal:



And so on...

We wish to determine if the patient's condition is stable, by measuring the difference between these various signals to see if any changes are occurring.



---

## **‘COMPARE’**

### **Problem**

We wish to obtain a measure of the ‘difference’ between two sampled-data signals.

### **Solution**

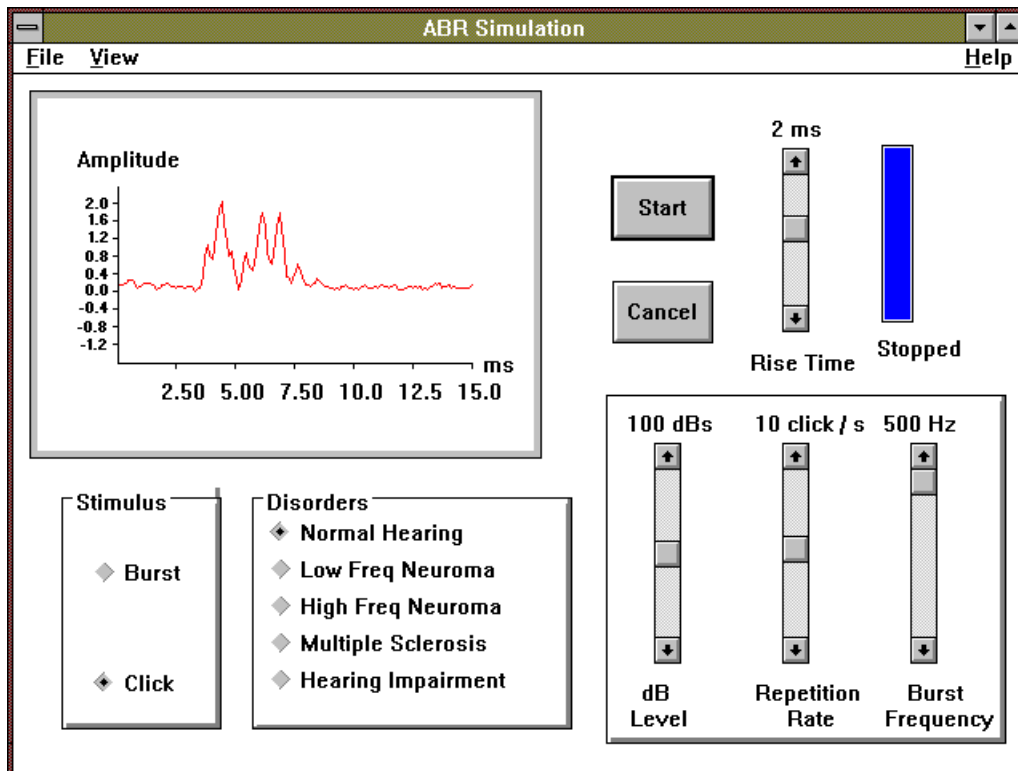
- Sum-of-squares
- Dynamic time warping
- ...

### **Notes / Comments**

### **Related patterns**

# 'MODEL'

## Example



From: Pont (1996).

---

## **‘MODEL’**

### **Example**

A steel pressing mill has been fitted with a vibration sensor (accelerometer). When a fault develops, this sensor will generate a voltage waveform, from which we wish to be able to determine the nature of the fault.

One way of doing this is to use an accurate model of the steel mill. When a vibration signature is detected, the parameters of the model will be adjusted so as to reproduce this signature. From the parameters used, it is possible identify the nature of the fault (if any).

### **Example**

Prior to implementing tax changes in an annual budget, these changes will be tested on a economic model, in order to predict their likely impact on different groups in the population.

### **Example**

The impact of continued production of ‘greenhouse gases’ on the planet’s climate can be predicted through the use of appropriate models.

---

## **‘MODEL’**

### **Problem**

Models are important components in many diagnostic and control applications

### **Solution**

There are two broad classes of possible solutions:

- Mathematical models  
(Particularly appropriate for linear, time-invariant systems?)
- Computer models  
(Useful when above conditions are not satisfied)

### **Notes / Comments**

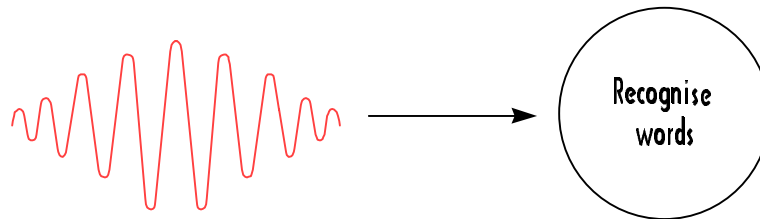
### **Related patterns**

---

## 'CLASSIFY'

### Example (Simple classify)

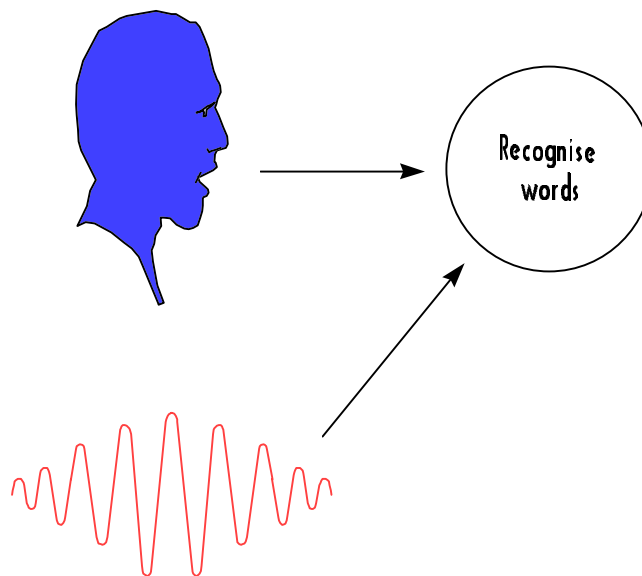
Speech recognition is a challenging problem. Generally, from the speech waveform alone (for example, the time-varying voltage waveform from a microphone), we aim to recognise the words spoken:



Neural networks have successfully been applied to this task.

### Example (Multiple classify)

In very noisy conditions, we may wish to attempt to use additional information from lip movements to improve the recognition performance:

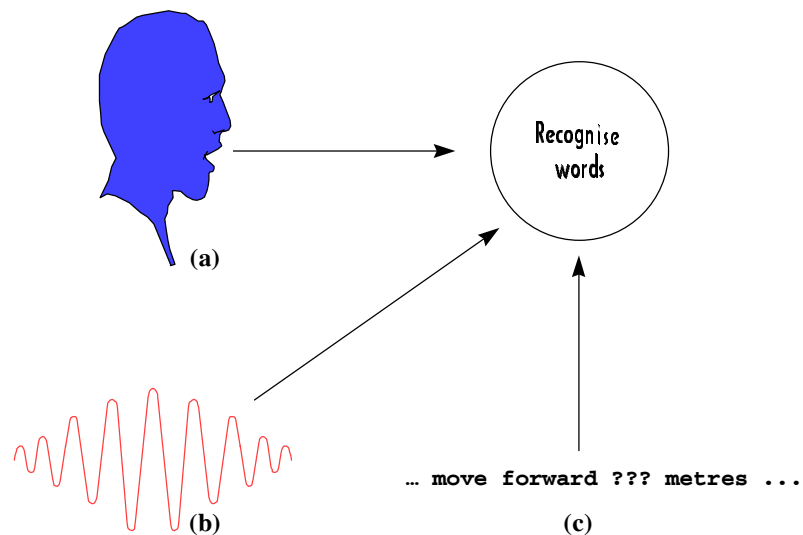


---

## 'CLASSIFY'

### Example (Heterogeneous classify)

In addition, context information might also be added:



The original HearSay system solved this type of problem using a blackboard architecture

---

## **‘CLASSIFY’**

### **Problem**

Classify is used to identify which of N groups a given signal S ‘belongs to’.

### **Solution**

- Statistical techniques
- Neural networks
- Neuro-fuzzy systems
- Blackboard systems
- ...

### **Notes / Comments**

### **Related patterns**

---

# 'CONTROL'

## Example

A stability augmentation system is required for a new range of wide-bodied jet aircraft. A schematic of the aircraft showing the relevant co-ordinates is shown in Figure 4.

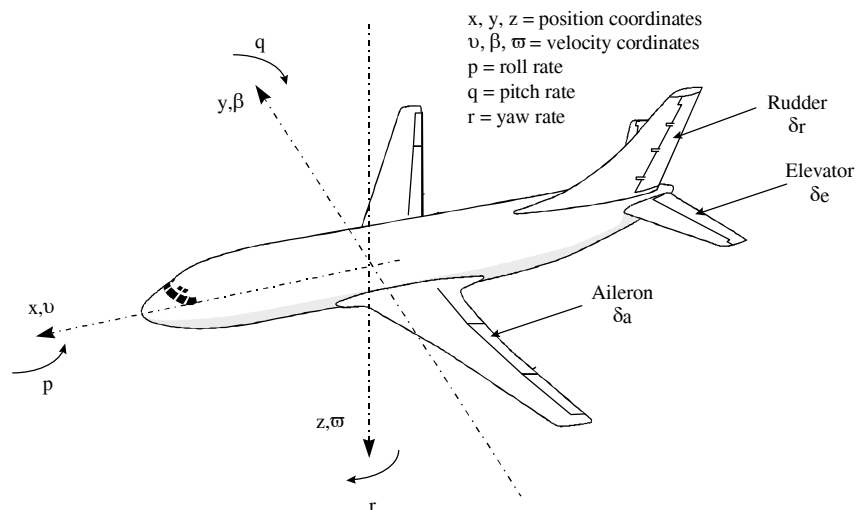


Figure 4: Definition of aircraft coordinates.

Like all swept-wing aircraft, it has a natural tendency to be lightly damped in one of its lateral modes of motion. At typical cruising speeds and altitudes, this mode is difficult for pilots to control without the assistance of a stability augmentation system (SAS).

## Problem

We often need to provide (feedback) control of a plant or process.

## Solution

- Lots of possible solutions (few of which could be attempted without experience)
- P, PI, PID control
- H-infinity
- Observer-based
- ...

## Notes / Comments

---

## CASE STUDY: MEDICAL DECISION SUPPORT (EMG)

### Background

Muscle contains groups of many fibres, which are cylindrical in shape and vary in length and width. Each group of muscle fibres is connected to a nerve fibre and the whole complex is known as motor unit (MU). Diagnosis of problems in the MU generally involves study of the electromyograph (EMG), a recording of electrical activity in the muscle during forceful contraction. Examination of the EMG from a normal human muscles reveals the presence of mainly bi and triphasic MU action potentials (MUAPs) with under ten per cent of MUAPs being polyphasic in shape (see Figure 5).

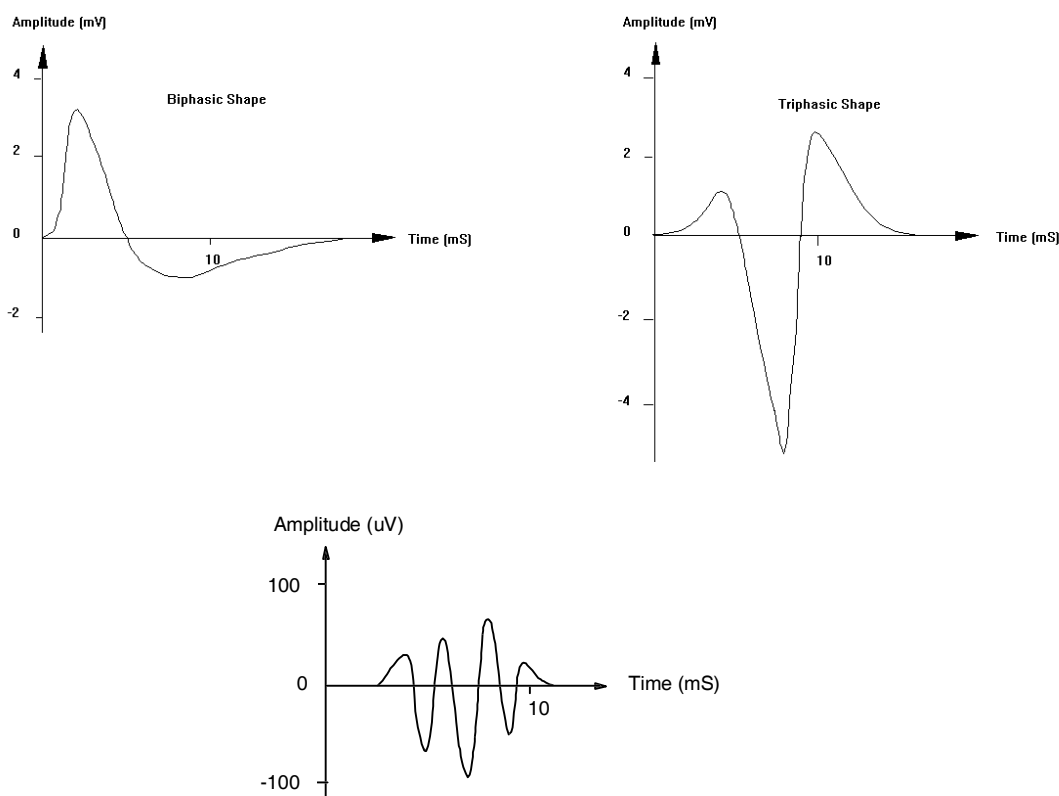


Figure 5: Examples of motor unit action potentials (MUAPs).  
(a) biphasic, (b) triphasic, and (c) polyphasic. See text for details.

---

## CASE STUDY: **MEDICAL DECISION SUPPORT (EMG)**

Because the same damage to muscle fibres and nerve tissue can be present in a range of diseases, identification of a particular disease a complex task. The DSS is intended to assist with this process. Specifically, the aim of the system outlined in this article was to distinguish between patients with neuropathies (diseases of the nerves), myopathies (diseases of the muscles), and patients (or controls) with no such diseases.

In a myopathic EMG signal the MUAPs are usually shortened in duration and smaller in amplitude than normal potentials and the percentage of polyphasic potentials is higher than the norm. The EMG signal is full at maximum voluntary contraction (MVC), but strength is reduced. In a neuropathic EMG signal the MUAPs are usually longer in duration than normal, and have larger amplitudes and a higher percentage of polyphasic potentials.

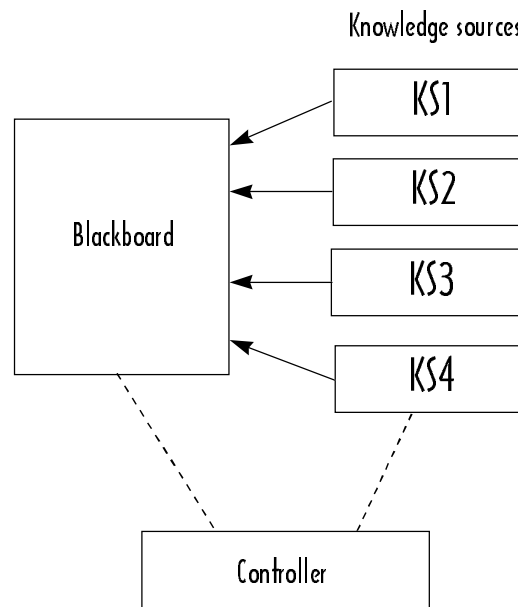
In addition to distinguishing neuropathies and myopathies, other possibilities must be considered. For example, myopathies must be distinguished from myositis (acute inflammation of muscle fibres): this can be done using an additional biochemical test measuring the level of blood creatine phosphokinase (CPK): if this is found to be grossly raised, this would suggest myositis.

---

## CASE STUDY: MEDICAL DECISION SUPPORT (EMG)

### Selecting design patterns

- We use Input (to sample the signals)
- The EMG system aims to classify inputs
- We require to work with both 'signal' inputs (EMG waveform) and numerical inputs (test results), etc...
- The 'Heterogeneous Classify' pattern seems most appropriate.
- One possible implementation uses a blackboard architecture:

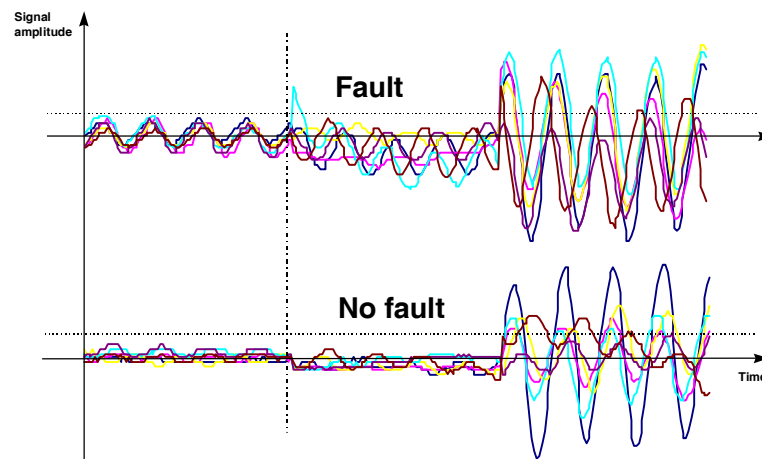


- The idea is that the 'pattern catalogue' guides the developer to the BB solution, and appropriate alternatives.

---

## CASE STUDY: ENGINE FAULT DIAGNOSIS

We wish to diagnose faults in an engine based on the vibration signature:



- Idea: 'Filter' (signal averaging) might help remove road noise, etc...
- Idea: 'Extract' might be sufficient (significant feature is the vibration amplitude...)

In the end 'Classify' was used (MLP neural network)

---

## CONCLUSIONS

- The talk has described some ‘work in progress’ looking at ways of improving the success of technical projects through the use of design patterns.
- Comments / suggestions on the ideas presented would be welcomed
- A copy of these OHPs is available on the WWW (<http://www.le.ac.uk/mjp9>)

---

## REFERENCES

- Alexander, C., Ishikawa, S., Silverstein, M. with Jacobson, M. Fisksdahl-King, I., Angel, S. (1977) "A Pattern Language", Oxford University Press, NY.
- Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P. and Stal, M. (1996) "Pattern-Oriented Software Architecture: A System of Patterns", John Wiley and Sons.
- Cunningham, W. and Beck, K. (1987) "Using Pattern Languages for Object-Oriented Programs", Proceedings of OOPSLA'87, Orlando, Florida.
- Gamma, E., Helm, R., Johnson, R. and Vlissides, J. (1995) "Design Patterns: Elements of Reusable Object-Oriented Software", Addison-Wesley, Reading, MA.
- Hatley, D.J. and Pirbhai, I.A. (1987) "*Strategies for Real-Time System Specification*," Dorset House.
- Pont, M.J. (1996) "Software Engineering with C++ and CASE Tools," Addison-Wesley.
- Pont, M.J., Moreale, E., Jones, N.B. and Mocroft, P. (in preparation) "Designing Technical Software: A case study in medical decision support".