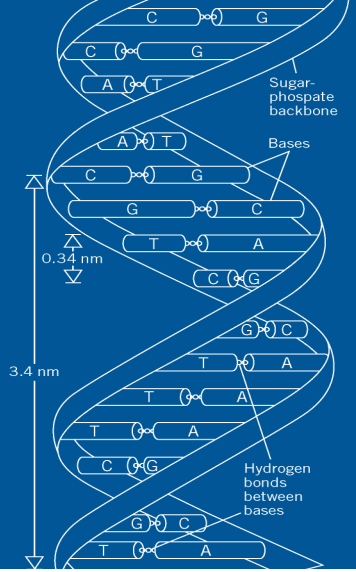


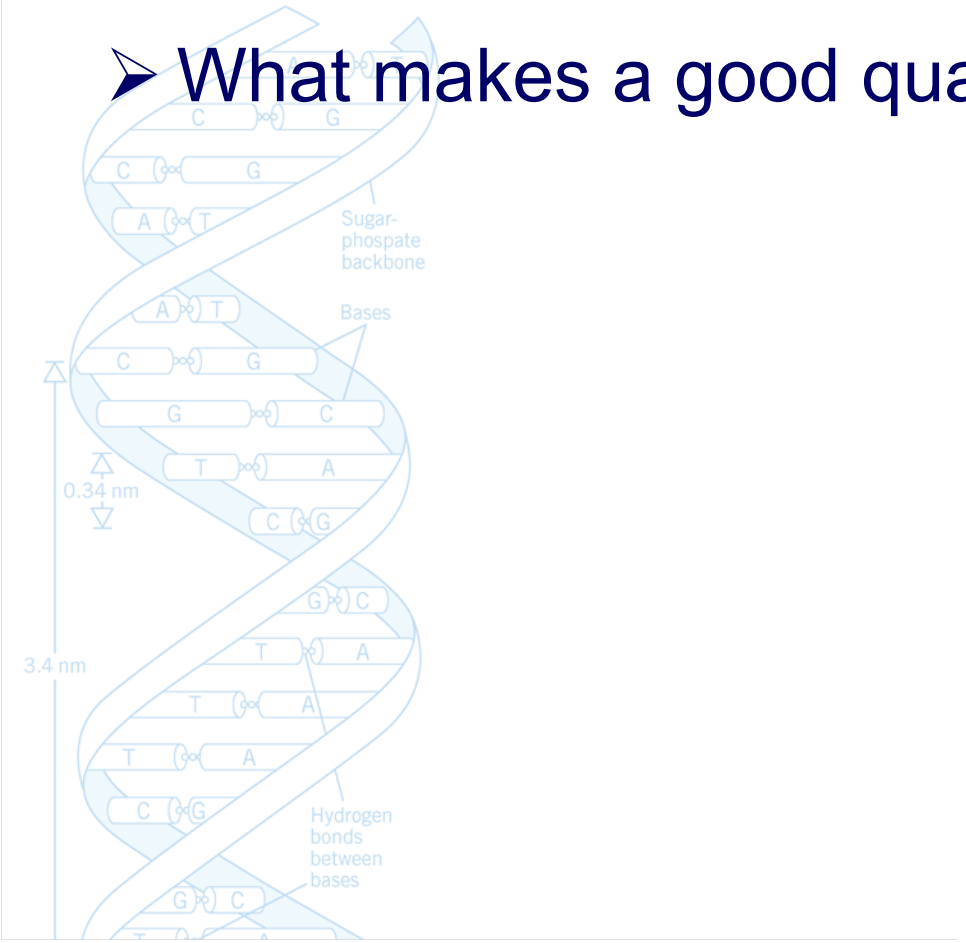
What makes a ‘good’ program?

Dean Grundy

PhUSE SDE – June 2011

Introduction

- What makes a good quality program?

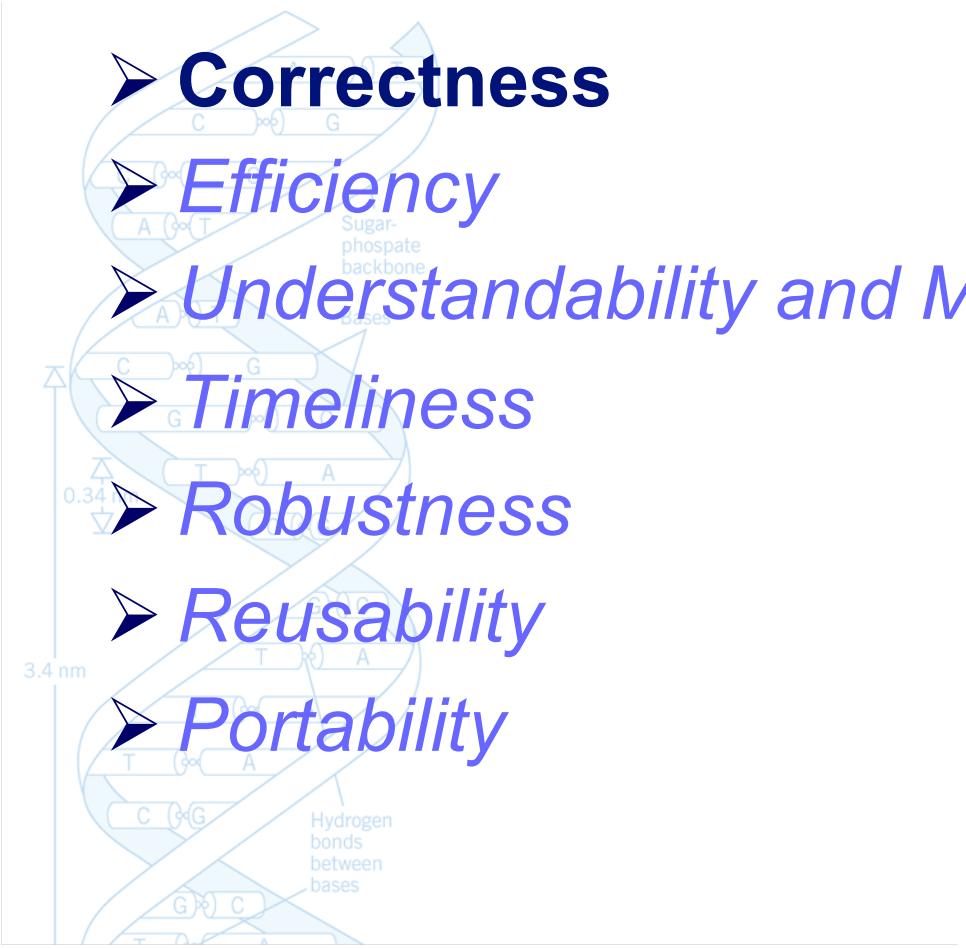


Software Quality Factors

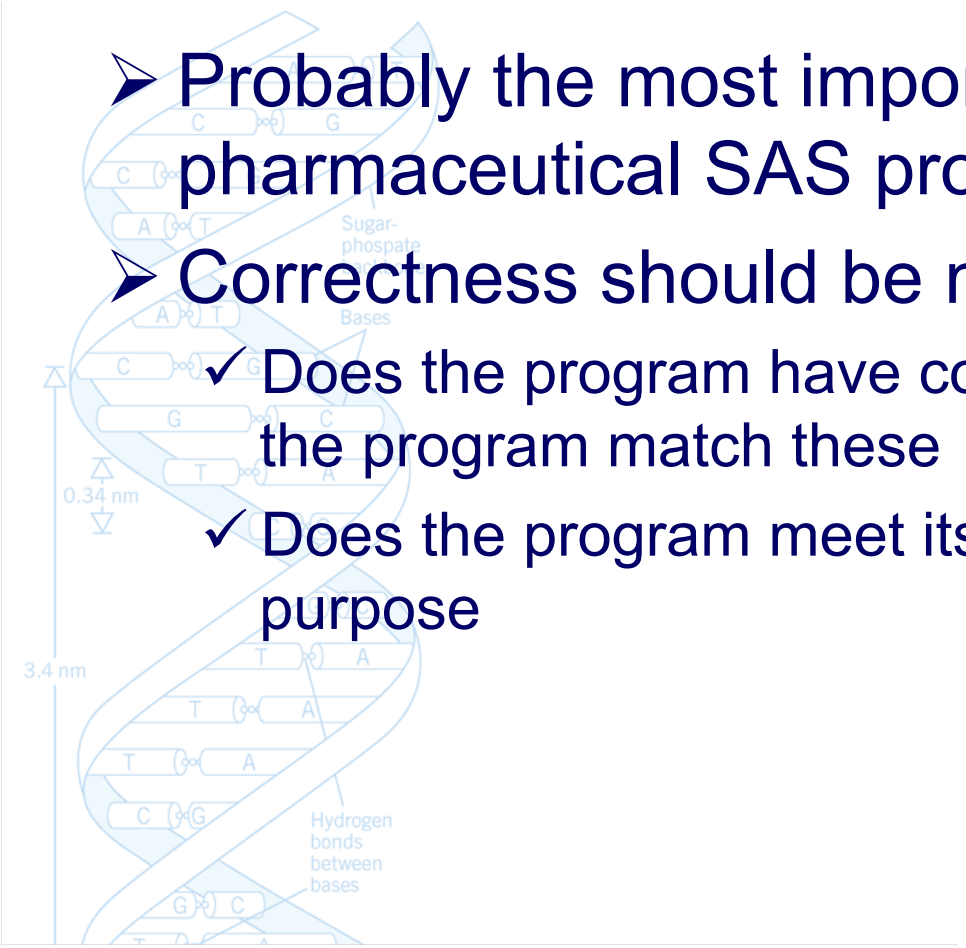
- Software quality factors are qualitative measures of determining quality
- Includes a wide range a factors
- This presentation will cover:
  - ✓ Correctness
  - ✓ Efficiency
  - ✓ Understandability and Maintainability
  - ✓ Timeliness
  - ✓ Robustness
  - ✓ Reusability
  - ✓ Portability

Software Quality Factors

- **Correctness**
- *Efficiency*
- *Understandability and Maintainability*
- *Timeliness*
- *Robustness*
- *Reusability*
- *Portability*



Correctness



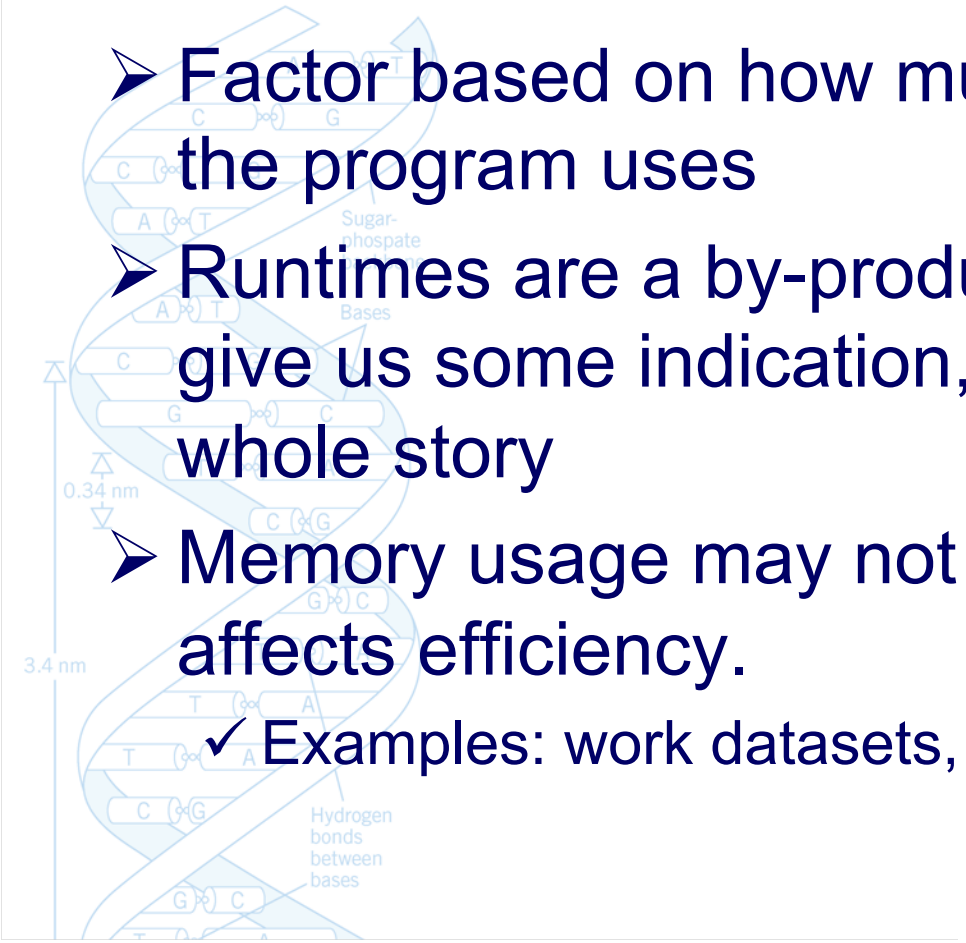
- Probably the most important factor for pharmaceutical SAS programming
- Correctness should be measured in two ways:
  - ✓ Does the program have correct specifications, and does the program match these
  - ✓ Does the program meet its requirements, and fulfil its purpose

Software Quality Factors

- *Correctness*
- **Efficiency**
- *Understandability and Maintainability*
- *Timeliness*
- *Robustness*
- *Reusability*
- *Portability*

Efficiency

- Factor based on how much system resources the program uses
- Runtimes are a by-product that can be used to give us some indication, but do not tell us the whole story
- Memory usage may not affect runtime, but affects efficiency.
  - ✓ Examples: work datasets, formats, macro variables



Software Quality Factors

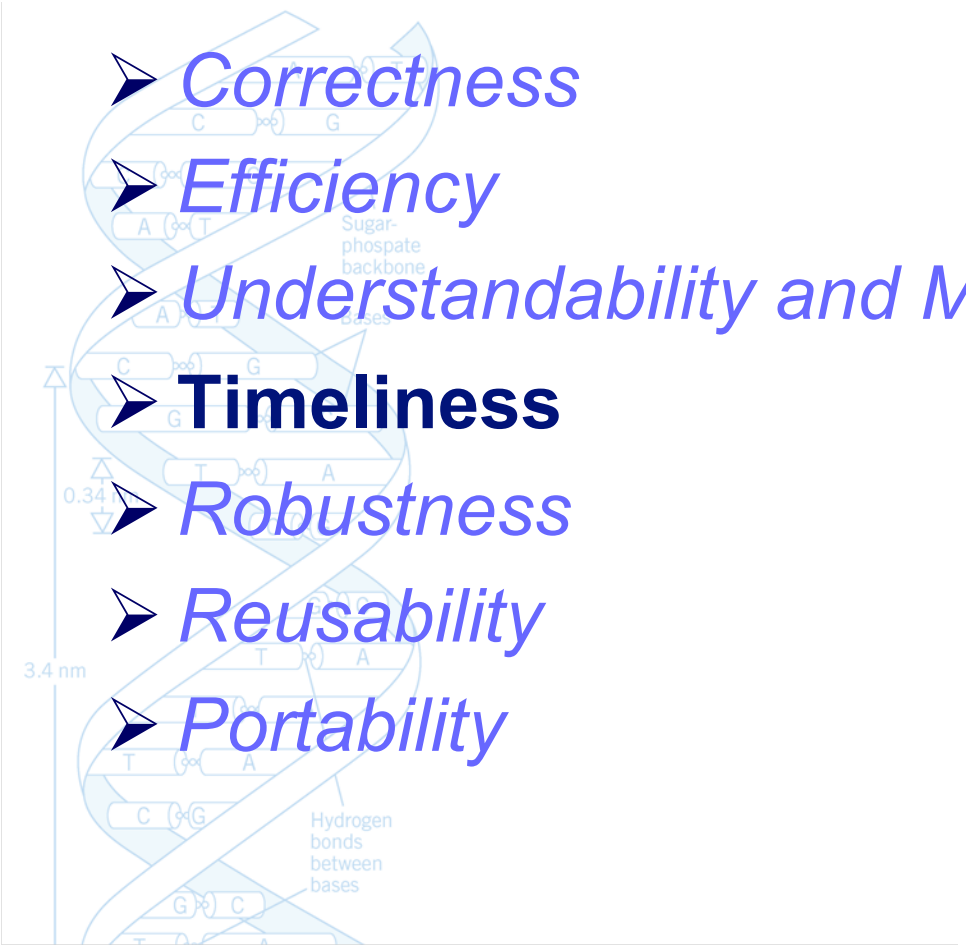
- *Correctness*
- *Efficiency*
- **Understandability and Maintainability**
- *Timeliness*
- *Robustness*
- *Reusability*
- *Portability*

Understandability & Maintainability

- Understandability is the measure of how easily the program code can be understood.
- Maintainability is concerned with how easily the program can be updated with any changes or enhancements.
- Strong links between the two, and compliment each other well to an extent, however there are differences
  - ✓ Example: If changes to one element result in edits throughout the program, the code can have low maintainability, but still easily understandable

Software Quality Factors

- *Correctness*
- *Efficiency*
- *Understandability and Maintainability*
- **Timeliness**
- *Robustness*
- *Reusability*
- *Portability*



Timeliness

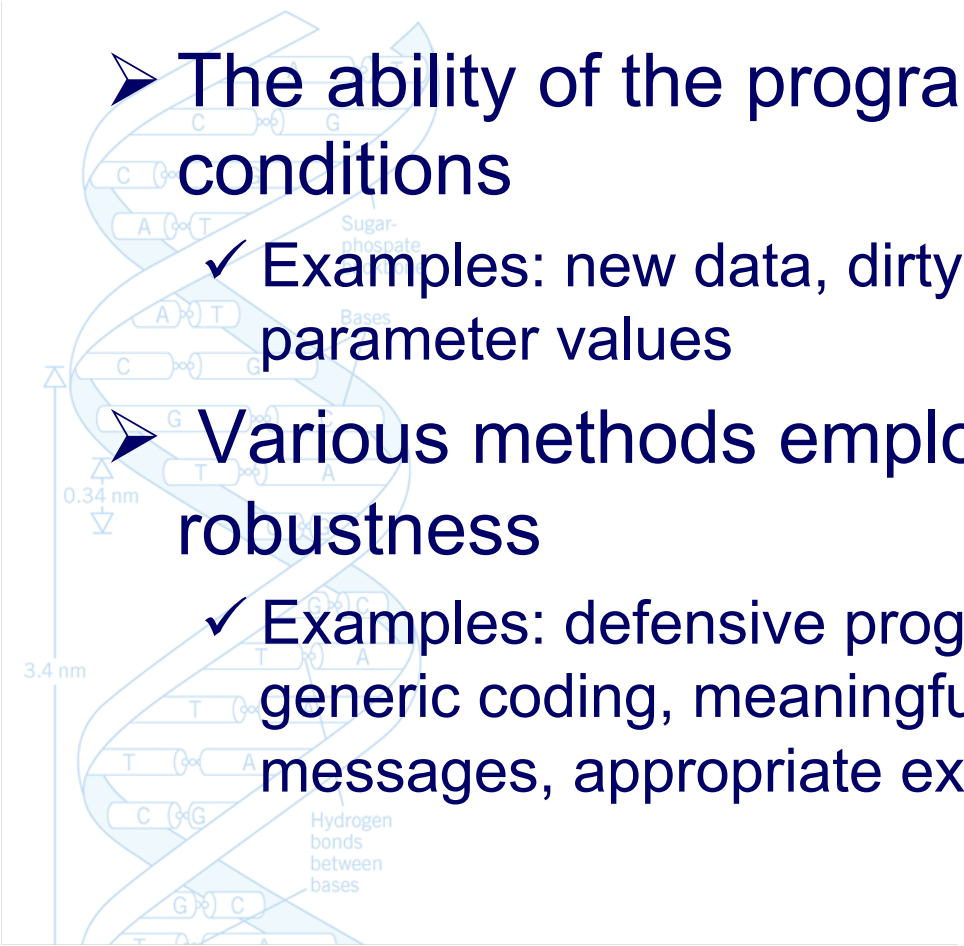
- Assesses whether the program is delivered on schedule
- Often not thought of as a factor, however elements to consider:
  - ✓ Is there significant benefit to delivering ahead of schedule?
  - ✓ Is the delivery date in line with when it is required?
- These elements can aid in determining the importance of timeliness

Software Quality Factors

- *Correctness*
- *Efficiency*
- *Understandability and Maintainability*
- *Timeliness*
- **Robustness**
- *Reusability*
- *Portability*

Robustness

- The ability of the program to handle unexpected conditions
  - ✓ Examples: new data, dirty data, unexpected macro parameter values
- Various methods employed to enhance robustness
  - ✓ Examples: defensive programming, dynamic and generic coding, meaningful user-defined error messages, appropriate exception handling

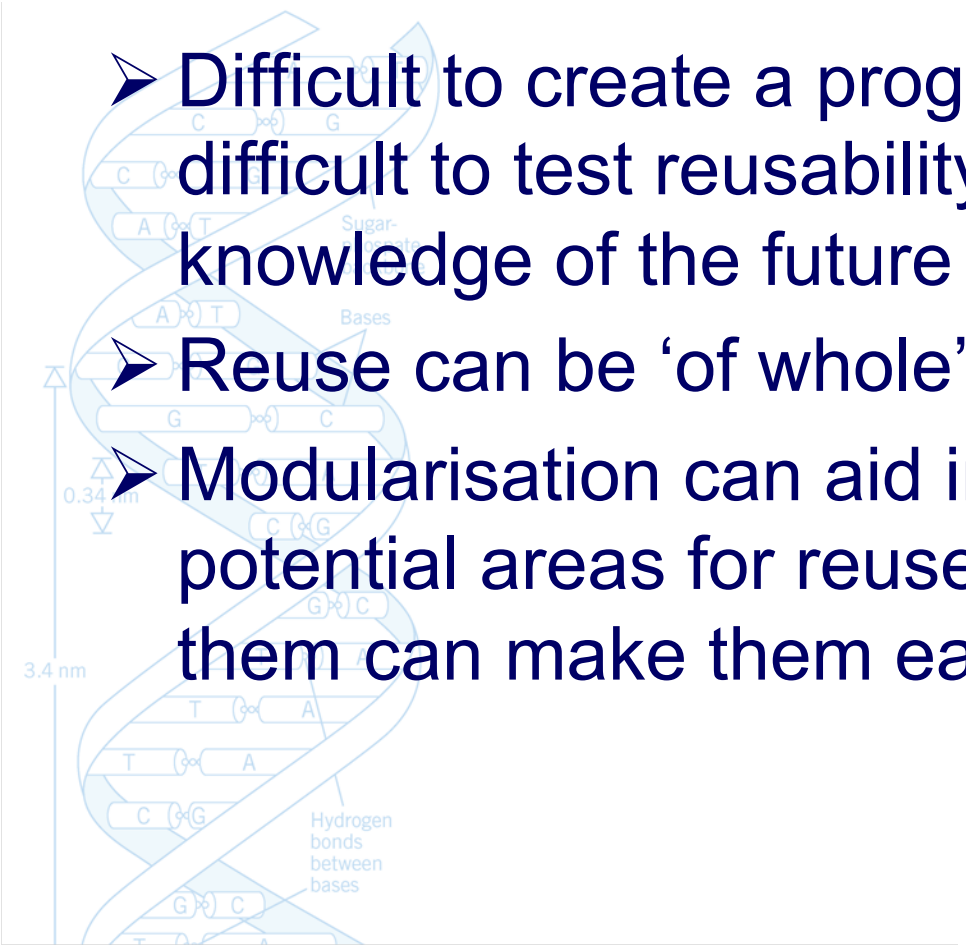


Software Quality Factors

- *Correctness*
- *Efficiency*
- *Understandability and Maintainability*
- *Timeliness*
- *Robustness*
- **Reusability**
- *Portability*

Reusability

- Difficult to create a program that is reusable, and difficult to test reusability, as it assumes knowledge of the future
- Reuse can be ‘of whole’ or ‘part of’ the program
- Modularisation can aid in reusability, identifying potential areas for reuse, and encapsulating them can make them easily transferable



Software Quality Factors

- *Correctness*
- *Efficiency*
- *Understandability and Maintainability*
- *Timeliness*
- *Robustness*
- *Reusability*
- **Portability**

Portability

- Is the code system/platform/SAS version dependent?
- Can be relevant in submitting programs to Health Authorities, or writing programs for another company.
- Even if the code is not completely stand-alone, encapsulating, segregating, or simply highlighting dependant code can ease the porting process

Testing

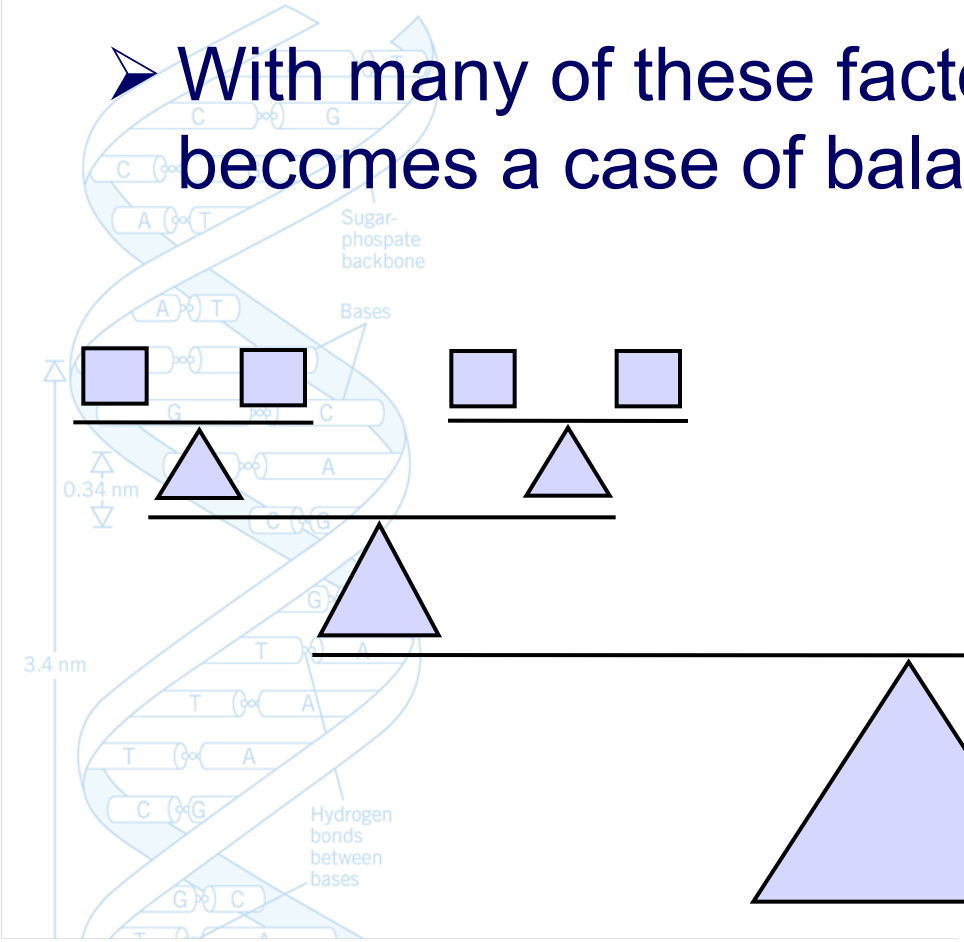
- Due to the qualitative nature of the factors, testing can be rather subjective
  - ✓ This is especially notable with factors such as maintainability, understandability, and reusability
  - ✓ Efficiency – do we have a baseline for how long it 'should' take, or how much system resources it 'should' use?
  - ✓ Results can very much depend on the assessor
- However tested, we are likely to get results which aid in improving the quality by investigating

Factor Interaction

- Many of the factors can have positive and negative affects on others
  - ✓ A highly efficient program is often not the most maintainable or understandable
  - ✓ A robust program may contain extra, potentially unnecessary, code to check for anomalies. This extra processing is an extra use of resource, decreasing efficiency
  - ✓ Large investment in factors such as reusability or portability may have negative impact on timeliness
  - ✓ Largely, overcompensation of one factor have detrimental effects on others

Balancing

- With many of these factors competing, it becomes a case of balancing these factors

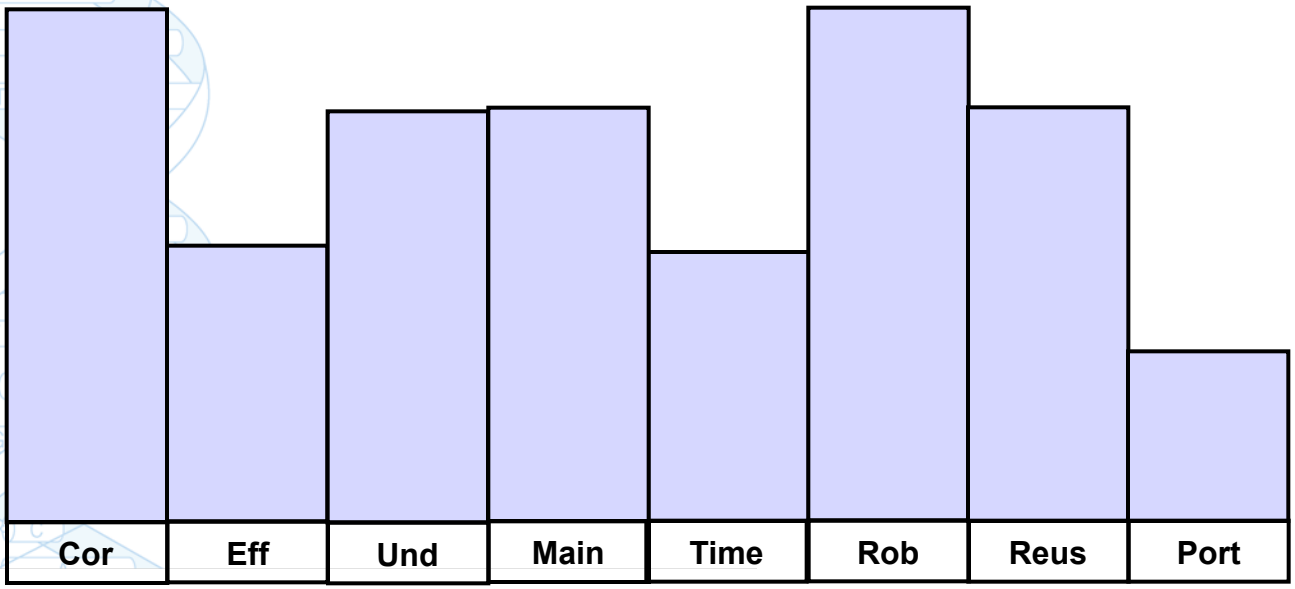


What makes a ‘good’ program?

- There is no single answer
- The right balance of the right factors for the right task in the right situation will yield a ‘good’ program
- Achieving this will have a greater chance if considered during planning – assessing what ‘quality’ means for this programming task.
  - ✓ ‘Program Quality Specification’
  - ✓ Specifying not ‘what’ to create, but ‘how’ to create it.

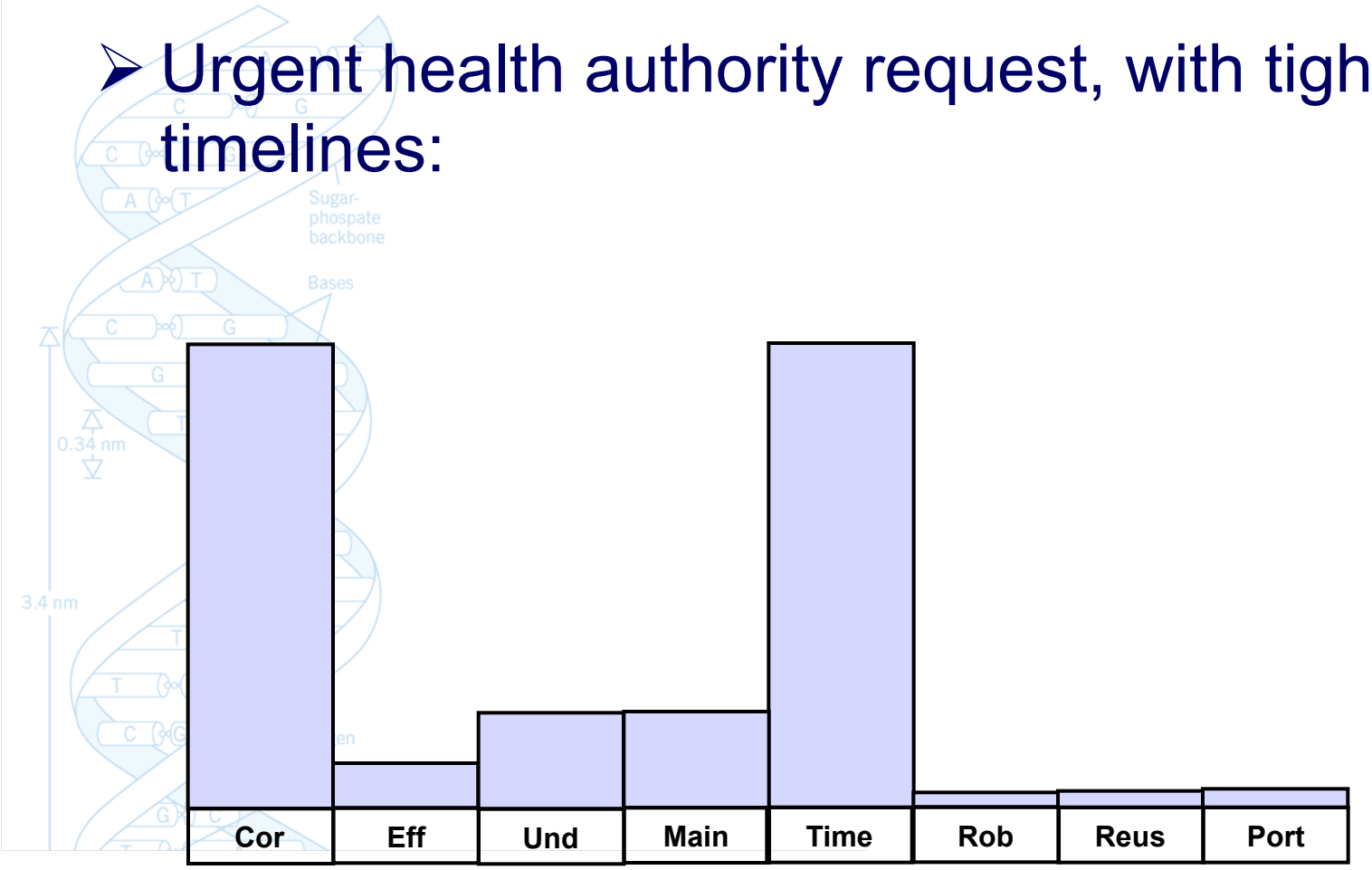
Case Studies

- Analysis dataset development, potentially could be utilised across many studies within the project. A possible SQF balance could be:



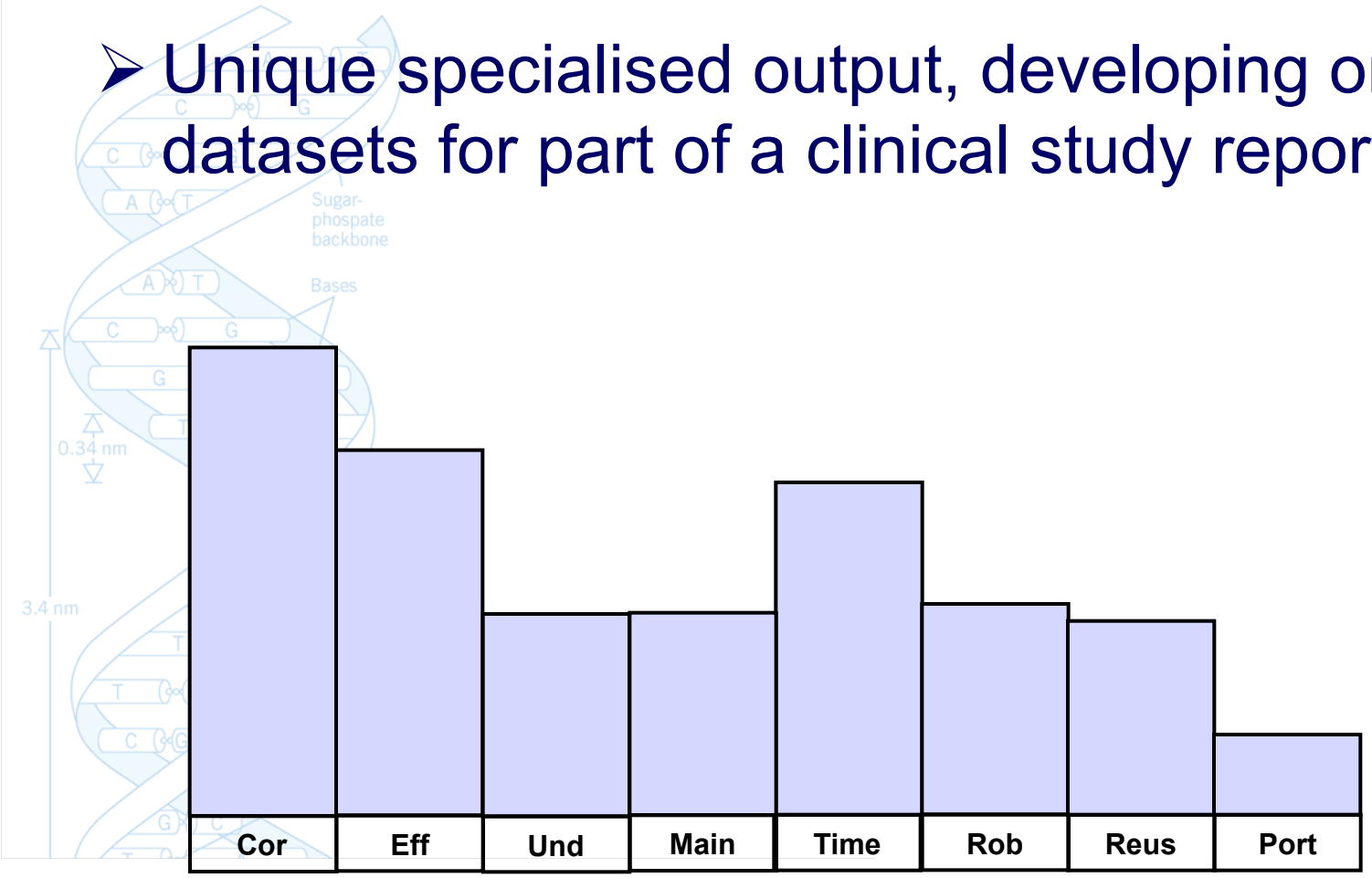
Case Studies

- Urgent health authority request, with tight timelines:



Case Studies

- Unique specialised output, developing on large datasets for part of a clinical study report



Questions

A
T
C
G
C
G
A
T
Sugar-
phospate
backbone
A
T
Bases
C
G
G
C
T
A
0.34 nm
C
G
G
C
3.4 nm
T
A
T
A
T
A
C
G
Hydrogen
bonds
between
bases
G
C