Computer Systems

DEFINITION

Computer systems
- Input/Output (I/O)
- Storage
- Processing

of

Software
- Instructions
- Data

INFORMATION & DATA

Number systems
- Binary
- Hexadecimal

Data and representation
- Analog v. digital
- Numeric data
- Instructions
- Images
- Sound
- Text

HARDWARE

Hardware
- Input/output devices

Storage and memory
- Secondary storage
- RAM
- ROM
- Virtual memory
- The CPU

Embedded systems

CPU architecture
- Von Neumann etc.
- Components
- Cache
- ALU
- Fetch-execute cycle
- Performance
- CISC v. RISC

Gates and circuits
- Boolean algebra
- Basic circuits

SOFTWARE

Software
- Operating systems
- Applications
- Utilities
- Virtual machines
- Open/closed source

Programming
- Low/high level lang's
- Assembly and LMC
- Translators/compilers/interpreters etc.
- IDE
- User Interfaces
- Standard algorithms

DATA EXCHANGE AND COMMUNICATION

Networks
- Infrastructure
- Internet & WWW
- Compression
- Encryption
- Security

Software applications
- Databases
- Spreadsheets

CULTURAL CONTEXT

Legal, environmental, moral & ethical issues
- Legal issues
- Moral & ethical issues
- Environmental issues

- Standards
- Reliability

TIME ➞

Programming

Handling data
- Variables
- Constants
- Data types
- Assignment operators
- Arithmetic operators
- Logical operators
- Relational operators
- Composite data types
- String manipulation
- File handling
- Databases
- Data structures
- Query languages

Control flow
- Sequence
- Selection
- Iteration

Depending on language being taught (some languages scope to statement blocks, not functions).

Modularity
- Subroutines
- Functions
- Variable scope
- Parameters (ref/val)
- Recursion

Encapsulation & OOP
- Classes
- Methods & properties
- Inheritance
- Polymorphism

Programming techniques and software development process
- Analysis
- Computational thinking
- Algorithmic thinking
- Problem solving
- Software development approaches
- Algorithm design
- User interface design
- Flowcharts
- Pseudocode
- Validation
- Development
- Documentation — incl. commenting
- Incremental testing
- Debugging
- Best practices* — e.g. variable/function naming, code formatting
- Testing
- Evaluation

Though the topics are presented ordered by the software development process, the order of teaching may differ to suit delivery of programming skills, e.g. debugging, incremental testing and standards may be taught earlier to instill good habits and best practices. Some topics may be re-visited throughout a course, e.g. computational thinking and problem solving.
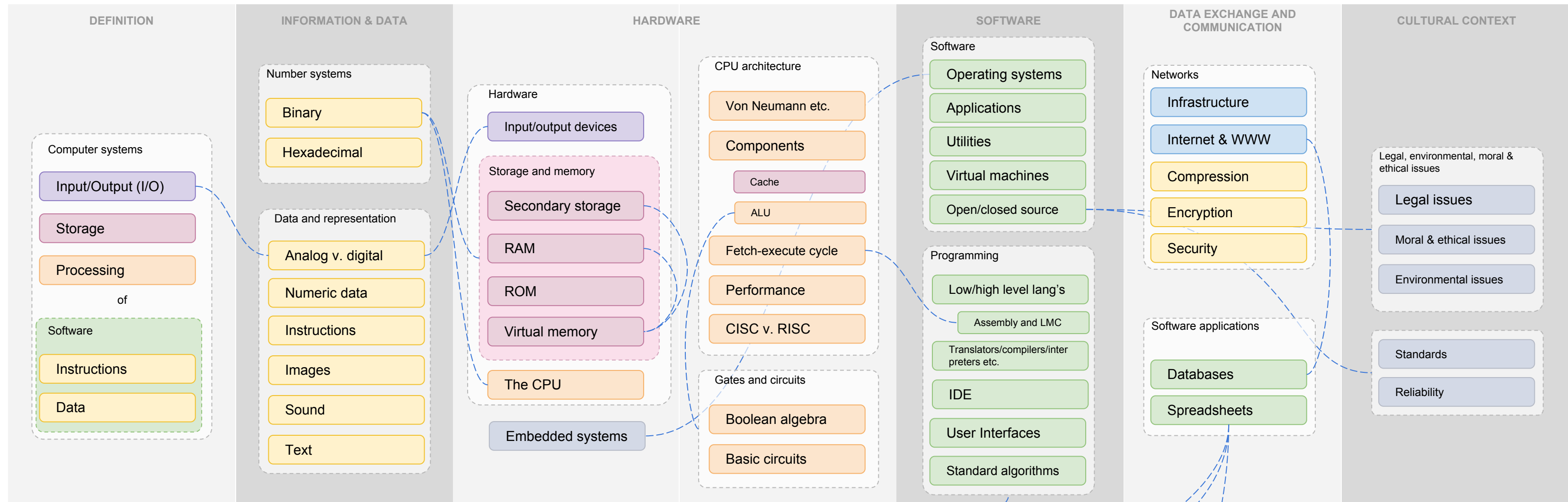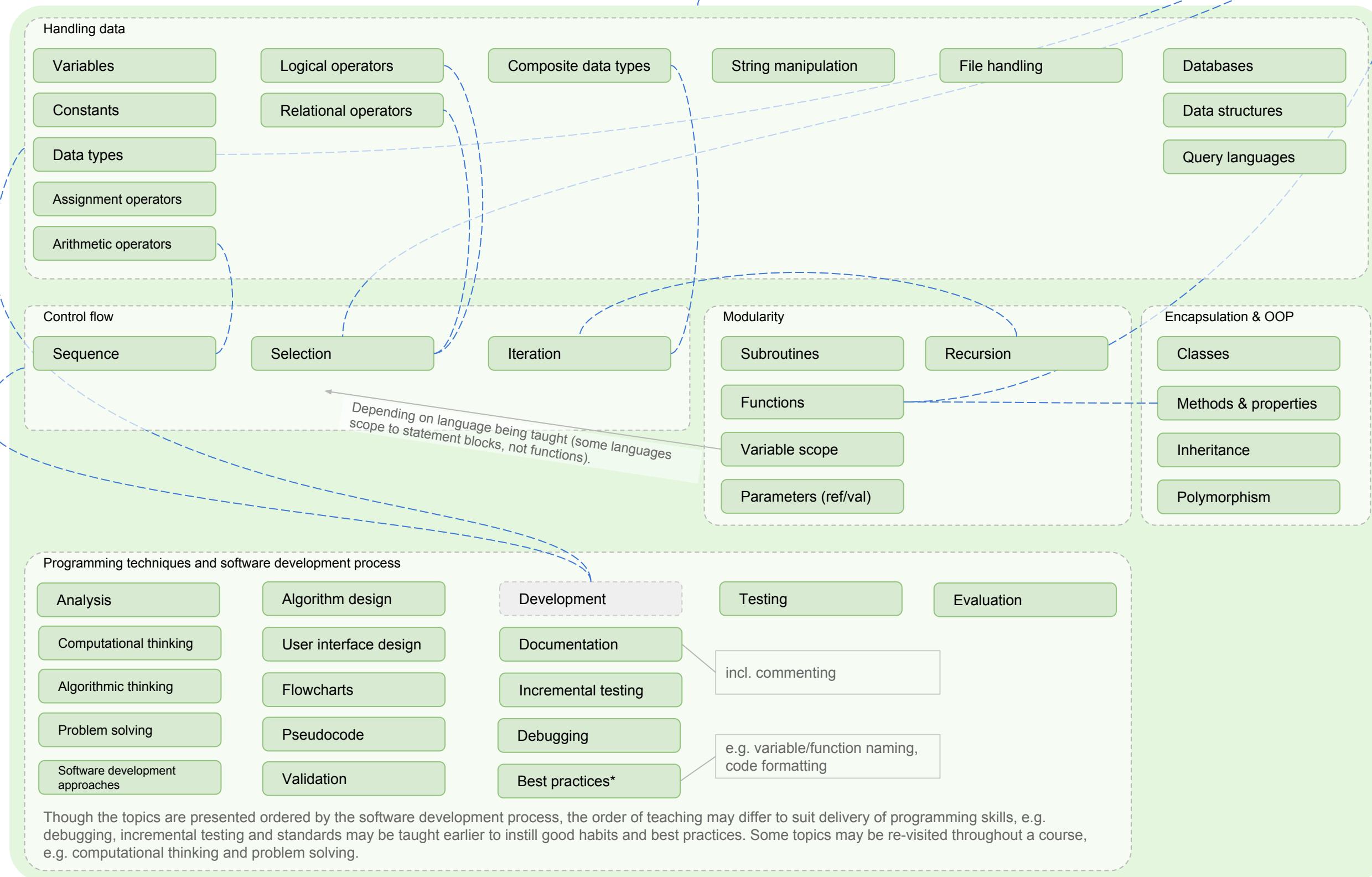
NOTES

The purpose of this document is to present key computing and computer science topics in such a way as make the relationships and links between concepts, knowledge and skills more clear. It is intended as a planning and teaching aid, particularly to help with:
- providing context for distinct units of work
- making clear how the content forms part of a cohesive whole
- linking topics to prior learning
- suggesting opportunities for re-visiting prior learning
- answering the question "why do I need to know this?".

By virtue of being organised to show logical relationships and a progression of complexity or abstraction, the document also suggests a possible order of content delivery (i.e. scheme of work).

Though not exhaustive, the document aims to cover most of the high level concepts in the UK KS3-5 curricula. Teachers will need to select what is relevant to the qualification and exam board being taught.

Colour designates explicit links between topics. Blue dashed lines designate implicit links. These are not exhaustive and are intended as examples only. Clarity is favoured over completeness.

Detail within key topics is only added when necessary to show explicit or implicit relationships between topics. e.g. components of the CPU.

SOURCES

Clarkson, M. (2013). GCSE Computing: The Unofficial Teacher's Guide. [e-book] Cambridge, UK: OCR. Available at:
http://www.ocr.org.uk/Images/139051-the-unofficial-teacher-s-guide-for-gcse-computing.pdf [Accessed 5 August 2016]

Dale N., and Lewis J., 2011. *Computer Science Illuminated*. 4th ed. Sudbury, MA: Jones and Bartlett.

Virnuls, A., 2016. *The representation of data links the whole of Computing*. TES Subject Genius Blog: Computing, [online] 14 April. Available at:
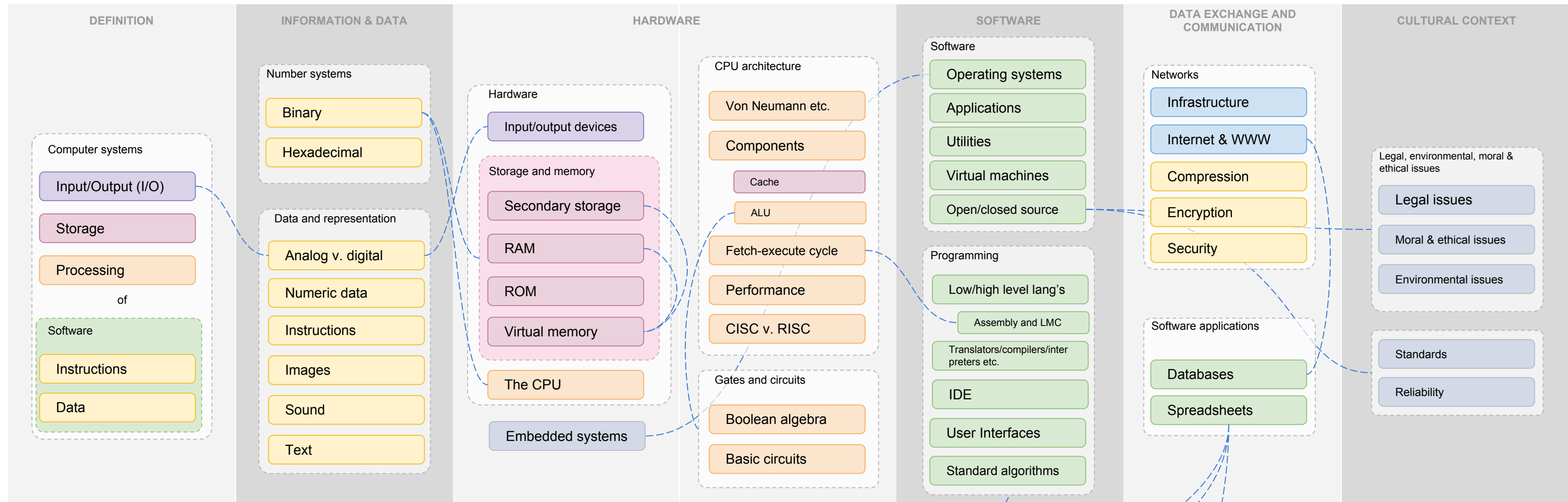https://www.tes.com/news/blog/representation-data-links-whole-computing [Accessed 5 August 2016]

Computer Systems

Number systems
- Binary
- Hexadecimal

Data and representation
- Analog v. digital
- Numeric data
- Instructions
- Images
- Sound
- Text

Computer systems
- Input/Output (I/O)
- Storage
- Processing

of

Software
- Instructions
- Data

Hardware
- Input/output devices

Storage and memory
- Secondary storage
- RAM
- ROM
- Virtual memory
- The CPU

Embedded systems

CPU architecture
- Von Neumann etc.
- Components
- Cache
- ALU
- Fetch-execute cycle
- Performance
- CISC v. RISC

Gates and circuits
- Boolean algebra
- Basic circuits

Software
- Operating systems
- Applications
- Utilities
- Virtual machines
- Open/closed source

Programming
- Low/high level lang's
- Assembly and LMC
- Translators/compilers/interpreters etc.
- IDE
- User Interfaces
- Standard algorithms

Networks
- Infrastructure
- Internet & WWW
- Compression
- Encryption
- Security

Software applications
- Databases
- Spreadsheets

Legal, environmental, moral & ethical issues
- Legal issues
- Moral & ethical issues
- Environmental issues

- Standards
- Reliability

TIME →

Programming

Handling data
- Variables
- Constants
- Data types
- Assignment operators
- Arithmetic operators
- Logical operators
- Relational operators
- Composite data types
- String manipulation
- File handling
- Databases
- Data structures
- Query languages

Control flow
- Sequence
- Selection
- Iteration

Depending on language being taught (some languages scope to statement blocks, not functions).

Modularity
- Subroutines
- Functions
- Variable scope
- Parameters (ref/val)
- Recursion

Encapsulation & OOP
- Classes
- Methods & properties
- Inheritance
- Polymorphism

Programming techniques and software development process
- Analysis
- Computational thinking
- Algorithmic thinking
- Problem solving
- Software development approaches
- Algorithm design
- User interface design
- Flowcharts
- Pseudocode
- Validation
- Development
- Documentation — incl. commenting
- Incremental testing
- Debugging
- Best practices* — e.g. variable/function naming, code formatting
- Testing
- Evaluation

Though the topics are presented ordered by the software development process, the order of teaching may differ to suit delivery of programming skills, e.g. debugging, incremental testing and standards may be taught earlier to instill good habits and best practices. Some topics may be re-visited throughout a course, e.g. computational thinking and problem solving.
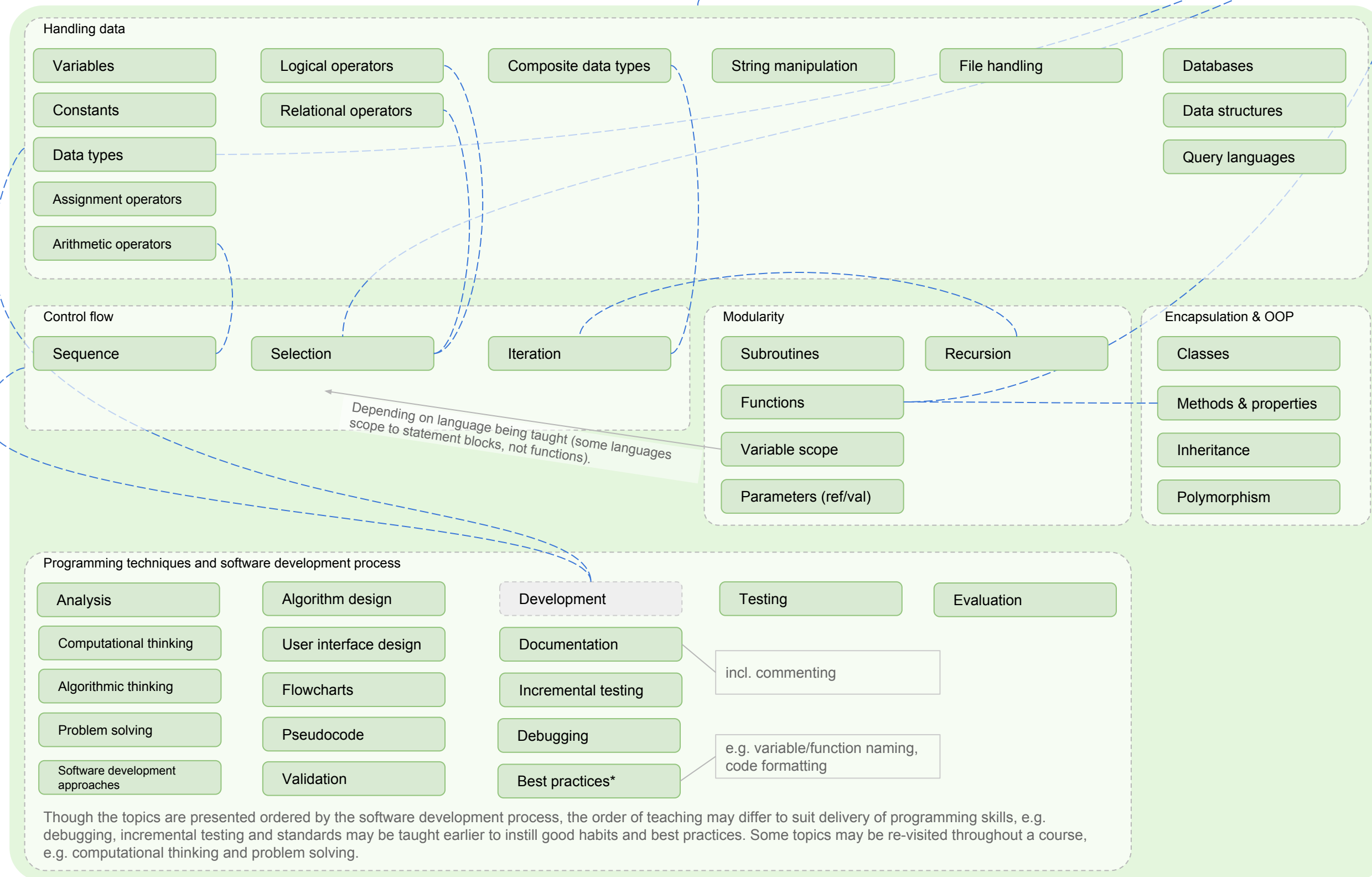
SOURCES
Clarkson, M. (2013). GCSE Computing: The Unofficial Teacher's Guide. [e-book] Cambridge, UK: OCR. Available at: http://www.ocr.org.uk/Images/139051-the-unofficial-teacher-s-guide-for-gcse-computing.pdf [Accessed 5 August 2016]

Dale N., and Lewis J., 2011. Computer Science Illuminated. 4th ed. Sudbury, MA: Jones and Bartlett.

Virnuls, A., 2016. The representation of data links the whole of Computing. TES Subject Genius Blog: Computing, [online] 14 April. Available at: https://www.tes.com/news/blog/representation-data-links-whole-computing [Accessed 5 August 2016]