Understanding IT concepts

Lecture Notes 2 for INF4280 – development of IT competence in organisations.

Jens Kaasbøll, Department of Informatics, 2009

Computer users need competence in three subject matter areas: representation of the domain, technology and tasks. Furthermore, for learning about IT, users need

- **Facts** about the applications and about who knows what
- Experimental **skills** for trial-error-success, playful exploration and metacognition, and social skills for making relationships with expertise.
- **Understanding** of concepts and principles, and why specific functionality is useful.

This lecture note addresses the concepts and principles from the latter category, and the main reason for learning them is that transfer of skills from one setting to another is easier when people understand the concepts and principles behind the operation of the technology (Bransford et al, 2000, p 9).

Computers and other IT technology are constructed on the basis of a few principles. Throughout half a century, more principles have been introduced for easing the design of software, and during the last three decades, these principles have also appeared in user applications. Some basic concepts will be introduced, and thereafter the process of learning them is considered.

Data

Types: Units of data with accompanying operations

Data in software appear in smaller and larger units, and for a unit of a specific type, there are accompanying operations. For example, a paragraph in a text processor document has operations like justifying, centering, aligning left or right; a cell in a spread sheet can be changed from number to date to text; and a file in an operating system can be copied and deleted.

Considering that users need to select operations which fulfill the needs of their tasks, they need to store data in a type which is appropriate for the processing they want. When needing to generate a table of data, they should choose a spread sheets when having to do calculations, while creating a table in a text processor will be a better option when formatting is more important.

The domain to be represented and the tasks to be carried out should determine the type of data selected. When scanning a document, users may choose whether to create a picture file or a format that allows also for storing characters which can be optically recognized by the computer application. The tasks to be carried out later might decide which option to choose; character recognition allows for searching through the text in the document, while a picture can be changed in contrast, colour, etc. Further, the picture format tiff is an uncompressed representation, leaving the picture exactly as scanned, while storing in the jpeg format compresses the file with some loss of detail, but at 5% of the storage space. This knowledge about two ways of representing may also be useful for working with scanning.

In general, the **type-instance** relation is known from programming. At the operating system level, it corresponds to that a file (instance) is of a specific format (type). In a text processor, a paragraph (instance) is of a specific style (type). While users normally would not change the definition of a particular file format, they may change the definition of a style in a text processor. When doing so, the paragraphs of that style will change accordingly, so that the type-instance relation is conserved, and the changes of the type has a cascading effect.

Types can be in a **generalisation-specialisation** relationship, meaning that the specialised type inherits properties from the general one. For example, the Heading 2 style in a text processor can be built on the Heading 1 style and inherits the paragraph formatting from the Heading 1. Whether changes on Heading 1 are propagated to Heading 2 and all its paragraph instances may vary according to the brand of the text processor.

Structures of data units

Data units of the same or of different types are organised into larger structures. Files in the operating system have hierarchical organisation, and cells in spread sheets are organised in a 2-dimensional grid. The structures allow creating wholes which are combined from parts, corresponding to the **aggregation** type of relationships.

Again, there are operations which can manipulate the structures, like inserting a new column in the spread sheet or moving a folder from one place in the hierarchy of the operating system to another.

Users also need to consider the match between task and data structure. For example, in order to produce a four pages folder to be printed, should one use a text processor or a presentation program?

Computers allow for creating links, references, shortcuts or whatever they are called in order to achieve two effects:

Functional dependency, meaning that when data is changed where stored, the changes are also accessible from where the link to the data goes. This principle ensures that data is stored one place, so that inconsistencies are prevented.

Breaking the structure, in the sense that from one place in a file, there is a reference which brings the user to anywhere else in the file or to another file, regardless of hierarchical, grid, sequential or other orders.

Type-instance and generalisation-specialisation relationships may all be considered variants of the functional dependency principle, where changes at one place cascades to those places which are linked to it and are supposed to be influenced by it.

Meta data

The properties of a file, which often can be found by right-clicking on the file symbol in the operating system, are data about the file, and this aboutness relation is normally called **meta**. Some of the meta data are functionally dependent on the contents of the file and its production process, while other can be set, like the access restrictions.

The layered architecture

A document from a text processor may be viewed in page break mode, as a continuous text, as a web page, etc. The contents and structure of the document remains, but the format is changed according to the way of viewing. One way of interpreting this phenomenon is to say

that the document can be separated in one contents and structure layer and one format layer, and that each of these can be changed independently of the other. In web page design, the structure and contents can be coded with html, while the layout can be set by Cascading Style Sheet codes.

Users who mix up the two layers are likely to do more work when changes have to be made than those who keep layout or format separate from structure and contents. For example, using a structure element like a blank paragraph in order to achieve a format effect, namely larger space between paragraphs, will have to change each paragraph. Paragraph formats, including space between paragraphs, can be set by the styles, which is a formatting tool.

In general, all data can be viewed and manipulated at many layers. For example, if a file is suspected to contain a virus, it can be opened by a Notepad or similar editor, which treats all data as characters, thus internal codes and user data are viewed as being of the same type.

While the deeper layers of the computer software is normally left for the programmers to deal with, having some insight into layers of the internet protocol may, eg, help users understand where connection problems reside.

The hardware layer has some principles which users need to cope with, since they have to grasp the difference between input and output. Some may also have understood that for example, the memory chip in a digital camera is also a general storage for data, so they can use it as a backup for their files when on vacation.

Of philosophical interest is the principle that data and programs are stored in the same way, so that programs can have other programs as their data and operate on them (Wikipedia: John von Neumann). This implies that the computer can change its own way of operation, and this property distinguishes computers from all other technology.

The learning of concepts

There are many models for learning concepts and theories, but very few have been applied for characterising how users learn IT concepts and principles. We will use one of these, the Actions-Process-Object model (Aharoni, 2000; Stamatova & Kaasbøll, 2007). Basically, this model says that people learn a new concept in three steps:

Action: The first step is performing some action, for example, referring to another cell in a spread sheet by typing a formula in one cell and clicking in another.

Process: Step 2 is when the learner can refer to the input and output of this action without actually carrying it out. In the example, the user would tell that in order to have one cell refer to another, one has to get the coordinates of the other cell into the formula.

Object: The final achievement is when the learner can refer to it as an object of its own and use it when talking about other phenomena. The concept would then have become reified. The spreadsheet user could say that cell-referencing is an ingredient in formulas.

When the users have built the concept like this, it is possible to learn other concepts which requires understanding of the first one, e.g., where cell-referencing is a part, or where cell-referencing is a special case. Let us assume that the goal is to learn the functional-dependency principle. Cell-referencing is a special case, and links between web-pages and table of contents in a document constitute other cases. In order to reach the action stage of functional dependency, the learners should consolidate the object stage of cell-referencing or another type of functional dependency. Thereafter, the learners need to get acquainted with

other types of the principle, e.g., the dependence between linked web pages or an automatically generated table of contents in a text processor. The process stage can be learnt after introducing the concept and discussing the similarities and differences between the various types of functional dependency relations. The object stage, where functional dependency should be compared with other types of relations, constitutes an advanced understanding that may be above the level needed for the common user, but should be the level of the teacher.

Beware that the action-process-object model is a model of learning, and not of teaching. Learners often do not understand the principles and concepts just from playing around on their own. Having the teacher first explain the concept and then demonstrate it, before the learner carries out the action in the first place, is in general a more effective way of teaching, according to the Gagné-Briggs theory of instructional design (Blanchard & Thacker, 2007, 198-201).

Incomplete reification - Interference

Some concepts, like the generalisation-specialisation relation between paragraph styles, might be quite far from previous experience, so few users will get to the object level where they can talk about is and discuss its benefits for their tasks. Cell-referencing seems simpler in the sense that most people seem to reach the object-level. Regardless of this achievement, people also systematically make mistakes, for example where the cell-reference concept interferes with Copy-Paste.

Assume that people working with spreadsheets for producing graphs calculate, by formulas and cell-referencing, one row of numbers D1-D12 based on the A-C rows. F and G constitute the bases for calculating H1-H12. The aim is to make one graph where the D and H rows are compared. Then they copy D1-D12 into K1-K12 and the H-row into the L-row, giving a matrix of K1-L12 from which the graph can be produced. Trouble appears if they have to correct B2, since K2 has no cell-reference to D2.

Knowing that the K-row should contain the same numbers as the D-row, the users choose the Copy command, instead of typing the formula =D1 into K1 and copy this formula to the right. The latter option would have kept the cell-reference from B2 through D2 and K2 into the graph. What is observed is that the Copy-Paste concept interferes with cell-referencing, since the effect of the reference in this case should be a copy of the value without any other calculation. The explanation of this commonly seen practice is probably that the Copy-Paste is more firmly reified than the cell-referencing principle.

References

Aharoni, D. (2000) Cogito, ergo sum! Cognitive processes of students dealing with data structures. *SIGCSE Bulletin* 32(1), 26 – 30.

Blanchard & Thacker (2007) *Effective Training. Systems, Strategies, and Practices.* (Third edition) Prentice-Hall.

Bransford, J.D., Brown, A.L. & Cocking, R.R. (2000) *How People Learn: Brain, Mind, Experience, and School*. National Academy Press, Washington, D.C.

Stamatova, E. and Kaasbøll, J. (2007) Users' Learning of Principles of Computer Operations. *Issues in Informing Science and Information Technology*. 4

Wikipedia (2009) *John von Neumann*. Downloaded from http://en.wikipedia.org/wiki/John_von_Neumann 26 Aug 2009