# Axioms for SNABOK, a System and Network Administration Body of Knowledge: Missing Link stage of Ontology Process

Jan A. Bergstra*

April 5, 2004

**Abstract**

SNABOK, a system an network administration body of knowledge, is coined as an attempt to complement the well-known and quite influential SWEBOK (software engineering body of knowledge) for the less prominent but equally critical profession of system and network administration. General remarks concerning the scope and limits of a SNABOK are discussed and cast in terms of 'axioms'.

Preliminary to a Stoneman, Ironman or Steelman version which should eventually emerge, Missing Link work expresses individual views. SNABOK axiomatization constitutes a part of the Missing Link documentation of SNABOK development.

## Contents



*University of Amsterdam, Programming Research Group, janb@science.uva.nl & Utrecht University, Department of Philosophy, Applied Logic Group, janb@phil.uu.nl. This work is done in connection with the MSc program 'Systeem en Netwerk Beheer' run by the University of Amsterdam in cooperation with the Hogeschool van Amsterdam.

# 1 Introduction

Much of the effort on a SNABOK (System and Network Administration Body of Knowledge) project can only be a copy of the well-known SWEBOK (Software Engineering Body of Knowledge) project as co-edited by Pierre Bourque and Robert Dupuis from Montreal (Canada). Indeed splitting up the subject into knowledge areas (KA's) is an obvious necessity. Further the preliminary SWEBOK explains that SWEBOK is based on computer science but covers different ground, and just as well SNABOK is based on computer science while it will cover different ground. A critical difference, however, is that the focus on administration prevents one from inheriting the principles from the engineering disciplines in general. The focus is on making existing software plus hardware artifacts

available for use to other users. This also involves the selection, configuration or composition, installation, the provision of instructions for appropriate use, and the operational integration of relevant artifacts given the tasks that a user or user group intends to perform.

## 1.1 Jargon: SNA, SNAC, SNAC's, SNA's, SNAT and SNAT's

A SNA is a system and network administrator; SNA in general refers to System and Network Administration. The plural of SNA is SNA's and a team working on SNA and consisting of SNA's at least is a SNAT. The plural of SNAT is SNAT's. Often a SNAT is referred to as a systems group but that phrase will not be used here. Thus a SNA works as a SNA member of a SNAT, often together with other SNA's and more often than not without much contact with the SNA's of SNAT's that support different units of the same organization or that are placed inside entirely different organizations. A competence characteristic to SNA is a SNAC, with plural SNAC's. Some SNAC's may be too specific for inclusion and description in a SNABOK. At this level of pedantry one may wonder whether besides a SNABOK a SNABOC covering the central SNAC's is needed as well. Currently that distinction will not be made but it may be of future relevance.[1]

Similarly we will use when needed the corresponding jargon for software engineering: A SWE does SWE, may be as a member of a SWET in cooperation with other SWE's, and perhaps in competition with other SWET's elsewhere. (S)he masters several important SWEC's and may be a certified expert on a subset of these.

### 1.1.1 One or more SNAB'sOK

Writing SNAB'sOK for the plural of SNABOK the following can be noticed: BOK projects are popular. E.g. the SA-BOK project by the SysAdmin Group aims at a formal body of knowledge for system administration, though based on a single company production process managed by Geoff Halprin. The project seems to have been completed in 1999. Under the assumption that SNABOK is a good idea, it becomes the issue to produce the best or the definitive, or at least a useful SNABOK among competition of other such projects. This leads to the viewpoint that many different SNAB'sOK can be imagined and that an axiomatic approach may be used to single out a fraction of the (potential) SNA'sBOK on which one intends to focus. Presenting such axioms is the objective of this paper. I begin with some axioms on X-B'sOK for any topic X in general.

**Axiom 0 for B'sOK (axbok0, Knowledge architecture axiom):** A BOK contains general information about the structure of knowledge for a subject. The detailed form of that knowledge need not be contained in the BOK. Thus a BOK is a knowledge architecture rather than a knowledge collection.[2]

[1] In the Netherlands, at the level of polytechnics (HBO for 'Hoger Beroeps Onderwijs' in Dutch), in the context of curriculum development the distinction between knowledge and competences is considered essential nowadays. This is in contrast with the situation at the universities, which are mainly focussing on knowledge, while the polytechnics (Hogescholen) take competences as the structuring concept. The technical universities may be placed in the middle but they don't use the competence oriented terminology.

[2] In the terminology of SWEBOK, a guide to the SWEBOK is produced rather than the SWEBOK itself.

**Axiom 1 for B'sOK (axbok1,Teaching and certification focus axiom):** Having an X-BOK available is supposed to be essential for teaching X and for the certification of the knowledge of those who have been taught X.

**Axiom 2 for B'sOK (axbok2, Implementation freedom axiom):** The abstraction to a knowledge architecture must leave room for differences in style and emphasis for teachers and students, it should also provide a form of technology independence.

**Axiom 3 for B'sOK (axbok3, Evolution axiom):** an X-BOK which is not maintained well cannot survive. Evolution of the knowledge of X should be reflected in an evolution of the architecture of this knowledge as given by a BOK.

### 1.2 A Missing Link Stage

The SWEBOK project went via a Stoneman version, and a subsequent Ironman version to a Steelman. Of course this is a good strategy, which should be followed. However, it may be relevant to add a preparatory phase, here called the Missing Link Stage (MLS) where more individual viewpoints and strategies are formulated and produced as an input. This prevents the risk that systematic consensus formation removes all creative or individualistic views that may connect the project with the research and the activities of individual people and groups. MLS documents (MLSD's) contain single author (or small team) consistent views on the subject matter of SNABOK. The views put forward in different MLSD's need not be consistent because the subsequent amalgamation process will filter out the options for effective consensus. Having clear missing link phase documentation available throughout the process allows individual participants to maintain their own views irrespective of the necessary compromise for consensus that may be needed for concerted action.

Only if a substantial number of MLSD's has been accumulated the construction of a team that may propose a Stoneman version can be considered. Each MLSD should preferably be augmented with a number of subsequent local field studies reporting on the state of affairs concerning various SNAT's in the working environment of the MLSD author(s).

Each MLSD may go through different stages and may have a local regional or even a national bias that suits its author. If different authors agree sufficiently they may merge their MLSD's thereby arriving at an MLSD version with more authors and with increased support.

### 1.3 A critique on SWEBOK

Reading through the trial version of the SWEBOK document from 2001, it seems to be the case that the meaning of the concept of software (or more precisely computer software) is taken for granted. At the same time it is explained in terms of concepts and principles what a professional is and what a body of knowledge should be. This is unsatisfactory to some extent. So in this section I will propose some definitions which I consider missing in SWEBOK, and which should in some form or another be addressed in SNABOK.

First of all the distinction between hardware and software requires some explanation. It is insufficient to suggest that software refers to bit sequences and hardware to atoms, crystals, fluids and molecules and so on. If that were true software engineering is about mathematical entities only, which in my view it is not. There is something physical to software just as well as there is to hardware. Suppose that systems are viewed as being composed from hardware and software. This by itself is a very non-trivial viewpoint as it fails to hold for human beings who can perform lots of tasks that can already be performed by computers and many other currently human tasks are expected to be performed by computers sooner or later, if computer science research and development successfully evolves.

### 1.3.1 The glass fiber definition of software

Now the following definition makes some sense: software is that part of a system that can be transported via a glass fiber. There is some flaw in this definition because the fiber might be used for energy transport, without any additional objective of information transport, and it seems hard to find a definition that cannot be compromised for any other purpose. If we choose a floppy disc rather than a glass fiber, the definition immediately fails because the floppy might for instance transport a biological virus which in my view is to be classified as hardware by all means. I take it for granted that no form of biological life can be transported via a glass fiber (though this requires some form of proof). That is the critical difference with artificial life. There are so many ways in which the physical constitution of the floppy itself may play an important role that it is best viewed as a means for transporting software via the transport of hardware, but not as a tool for the definition of software. Is it correct to claim that all software might be transported via a floppy disc? Apart from obvious storage limitations this is not the case. A floppy cannot reasonably be used for transporting real time data, whereas glass fiber can. Using the above definition real time measurement data are to be classified as software, however. The 'glass fiber definition' of software has no bias towards software for computer control, as it covers the software from the music industry just as well.

### 1.3.2 Digital or not?

There is no compelling reason that software must be of a digital nature, possessing a mathematically accurate description as a finite bit sequence. Analog signals may be passed through a glass fiber as well and perhaps also the qubits of a quantum computer, which defy the digital paradigm to some extent.

### 1.3.3 The bit stream abstraction and proactive software

It is reasonable to try to assign a meaning to software in technical terms. In principle for each signal emitted from a glass fiber connection its time of emission can be determined with information theoretic limits relating to optics and quantum mechanics imposing bounds on the precision of this temporal information. Forgetting about timing a piece of software (= a time interval with all

information about the transmission along a fiber), may be abstracted to a bit sequence. This abstraction is not obvious and it requires a significant amount of information theory and physics. For real time data the bit stream abstraction is pointless. It is very much up to the receiver of the software whether or not a bit stream abstraction makes sense.

Let us assume that the software part of a system splits in software that can be transported with arbitrary timing (i.e. a small shift in timing cannot enact a big difference in the behavior of the system importing the software) and other software for which the timing is vital. The first part of the software may be called proactive software (that will cover both control software and static data software), whereas the second part of the software constitutes interactive software (including the transport of real time data) .

JIT (just in time) delivery of proactive software shifts it in the direction of interactive software but to some extent the definitions still work.

### 1.3.4 Control software as a subcategory of proactive software

Proactive software may now be classified as either control software[3] or data software. The distinction is arbitrary, however, and many intermediate stages exist. Given a system and a theory about the role that a piece of proactive software plays in it, a piece of software may be considered control software but a the same time the same proactive software may be considered data software in an other system. In the absence of any explicit or implicit temporal information or constraint it is most plausible to view a piece of software as data. If it is control software somehow a temporal succession of events must be contained in its semantics.

The distinction between control software and static data software is fundamentally arbitrary: one is inclined to view the contents of a music track as static data software and the executable for a computer game as control software. This distinction is probably due to the fact that until recently authors and performers of music were perceived totally different from computer software writers, in terms of ambition, mind set and tooling. But that may change and so may the suggestion that music is static data software rather than control software. Indeed, the current perception of music seems to allow for very little interaction which suggest the absence of control software as an important constituent of the music. If music becomes interactive its descriptions will shift from data software to control software.

It seems valid to say that SWEBOK's concept of software is contained in the control software as a subcategory of proactive software, admitting JIT delivery when needed. At this point, however, an important decision must be made that remains implicit in the SWEBOK document.

[3] An theory of control software in the form of a code algebra based on the machine functions of [5] has been proposed in [1].

### 1.3.5 Programmed control software versus adaptively generated control software

There are at least two quite different classes of (computer) control software. The programmed control software and control software that has been generated or learned by some adaptive system. A neural network inside a robot may learn a task during an adaptive process where external forces indicate to the robot how to behave under certain conditions. The result of this adaptive process is a matrix of thresholds that may be transported via a glass fiber connection to another robot far away with the effect that after the transport the second robot is able to perform similar tasks as the first on has just learned. This indicates that the result of the adaptive process is software which can be imported in the second robot for enabling new behavior. This particular form of software may be considered proactive software, and depending on the nature of the task it is proactive control software or proactive data software. If all that happens is making the robot listen to music which it thereafter can reproduce the functionality is that of an acoustic recorder, usually a reason to refer to the software as data software rather than control software. As far as I can see the adaptively generated control software fails to fit the framework of SWEBOK, it being completely focussed on the engineering process for programmed control software.

### 1.3.6 Proposal: SWEBOK = pSWEBOK + aSWEBOK

I consider it perfectly reasonable to view adaptively generated control software computer software and to consider its manufacturing a branch of (computer) software engineering. If this is valid SWEBOK is missing out half of the playing field. The missing half may be of growing importance, however, and may sooner or later become the dominant style of engineering in this area. As a consequence I consider the SWEBOK framework to be based on a current technology, rather than on a fundamental analysis of concepts of definite value. Another way of looking at the matter is that SWEBOK is about programmed control software engineering, pSWEBOK say with aSWEBOK (focussing on adaptively generated control software) still to be developed, and potentially more important.

Still I hold that artificial intelligence has been delivering methods for producing (adaptively generated control) software and by that very reason provides a part of the technology of software engineering, with programming providing the other major part, which admittedly is currently commercially and technically more important. Whatever way these considerations go, I feel that a modern programmer is much better of viewing him/her self as a software engineer broadly conceived with pSWEBOK technology as his/her specialization because that minimizes the chance of software engineering becoming an obsolete profession overnight, simply due to the SWEBOK definition not being sufficiently technology independent.

### 1.3.7 Is the programmer a human being? And the computer?

The term computer stood for a person performing a calculation until the 1940's.[4] After the tasks performed by (human) computers in those days had been automated, the machines doing so were

[4]See: Steve Ditlea, Binary Choices and the Tao of Home Computers at www.atariarchives.org/deli.

called computers and the human profession of that name became outdated. The cause of this phenomenon is a lack of technology independence in the definition of the (human) computer. Of course the modern computer is just a better tool than the tools previously used by the human computer. There was no compelling reason to view the computer as a machine. Since the 'loss' of the computer as a person one needs the concept of a user even in cases where this is silly. The human computer used to be a tool user and so is the modern user of an electronic data processing device. Nowadays the successor of the human computer is the expert in computational science. It would still be preferable to be able to say: 'I am a computer' in my opinion.

Let us now focus on the programmer. It is possible though perhaps unlikely that the majority of programs will be generated by computers in the future. Certainly the step from what currently counts as a design to a program may be much more automated than it is today. In that phase of technology the designer becomes a programmer because he/she is the person who plays a crucial role in the process that leads to the existence of new (programmed control) software.[5] The SWEBOK split in program design and program construction with the programmer a specialist on program construction rather than design is a corollary of the ever present low appreciation of computer programming as a discipline. Taking a more technology independent perspective on programming and insisting that the phrase will not stand for a machine all of a sudden simply because its definition has been made technology dependent more than actually needed, one may provide a better guarantee that programming will refer to a human task as long as people are involved in the process.

### 1.3.8 Technology independent professional labels

Consider the following roles: judge, lawyer, politician, manager, nurse, physicist, mathematician, musician, artist, painter, sculptor, sportsman, financial consultant, veterinarian, management consultant, civil engineer or priest. Each of these roles has been made quite robust against changes in technology. These terms are remarkably technology independent defining roles within the social structure of people rather than within a technological context.

Considerable less technology independence has been achieved with the following profession descriptions: bank employee (probably doing something obsolete or nearly obsolete), typist (a service now limited to the happy few), printer (by default a peripheral device), punch card typist (obsolete), data entry typist (a task that can be automated in each case), type setter (now a program), computer (now a machine), programmer (failed to be a designer, or even better a software engineer), help desk worker (probably having a desk for him/her self at a call center while keeping the customer at a safe distance), ICT-consultant (fails to appreciate the human factor unfortunately), systems analyst (outdated jargon), software tester (now a software quality manager), SAP consultant (alive and kicking now but..), Unix specialist (still going strong), Fortran programmer (a profession of the past but his Cobol programming colleagues still have more work than expected), librarian (of course, but what is a library today), milkman (superseded by a combination of technologies), dust collector (recycling specialist!), miner (when will this be left to robots), beer brewer (refers to a whole brewing enterprise now), computer operator (quite outdated jargon, human proxy of a user, no human successor), ticket seller (use the internet), systems programmer (probably still in

[5] This argument has been discussed at length in [2].

business), data miner (googlist or googler in obscure newspeak).[6]

### 1.3.9 Conclusion comments on SWEBOK

SWEBOK takes the interpretation of software as programmed proactive control software admitting a digital representation for granted. That, however, unnecessarily introduces a technology dependence. For SNABOK both parts of the phrase: 'systems and network' and 'administration' will require equal care. There is a vast literature on administration, which is dominated by an interpretation of administration that puts control over human organizations in the center. This is not what seems to be meant in SNA, however. For SNA I conclude that the ontology of 'S & N' needs more attention than SWEBOK paid to the ontology of 'SW'. Below that aspect will be addressed in more detail. This leaves open the question to what extent general princples of administration inherit to SNA.

## 2 SNA, axioms for a human task

In a SNABOK, SNA should be put forward as a human task which is robust against foreseeable technological change, the mere appearance or disappearance of technology cannot have adverse consequence concerning the SNA profession at large. Of course the number of people involved may change just as the use of larger aircraft may lead to a need for fewer pilots. The SNA is responsible for the effective operation of a computer system or a group of such systems. As long as hardware and software are distinguished this responsibility covers both hardware and software. The SNA will be responsible for some amount of regular operation needed to guarantee continuous adequate operation of a system. Similarly (s)he will play a significant role in the design of system innovations, whenever either changing business or changing technology calls for such an innovation. Whenever a task that has been considered a characteristic part of SNA (like running a help-desk, distributing login names, fighting viruses and worms, or installing new operating system releases) disappears from the regular tasks of an SNA, SNA is redefined by deletion of the task (and probably of related tasks), and it may be extended elsewhere by including work that has become important for the main responsibility of SNA. This additional work may exploit a technical competence, but it may be just as well a people skill based competence.

**Axiom 0 for SNABOK (snax0, Human user support axiom):** SNA acts in favor of an organization or institution or of a unit of people and its members. If the SNA were the only member of that organization (s)he is a user rather than a SNA. It is the enabling role towards other users that primarily characterizes SNA rather than any listing of tasks ad competences.

**Axiom 1 for SNABOK (snax1, Human task axiom):** SNA is a human task that cannot be fully automated, by definition. Changes in technology may lead to changes in the tasks of SNA but

[6] In Dutch one has: pompbediende, telefoniste, treinconducteur, meteropnemer, ponstypiste, kassière, automatiseringsdeskundige, zetter, brugwachter, derde piloot. Each of there terms refer to (almost) outdated professions, invariably due to either changing technology or to a changing role of technology. In most of these cases the profession will not even survive in another form which could have been indicated with the same name if that had been chosen more carefully.

like the pilot survived the transition to jet engines, the SNA will outlast many forms of progress in system technology, in particular those changes that were invented in order to reduce the costs of system administration.

Currently and within a foreseeable future it is the case that systems consist of hardware and software. Using the classification of software presented in the previous section, the software involved falls into three categories: proactive software, which is either proactive control software or proactive data software and interactive software. Now it seems fair to assume that for the tasks of SNA it is immaterial to know or to assume that control software is programmed control software, because a complex system may use adaptively generated control software just as well. This leads to the next axiom of SNA.

**Axiom 2 for SNABOK (snax2, Black box view on control software):** SNA distinguishes software in two classes: proactive software and interactive software, where proactive software is split into two subclasses: control software and proactive data software. Interactive software is merely interactive data software. No further classification of control software is relevant for SNA. This may be considered a black box view on control software, as no inspection of the internal structure or working of the control software is called for. In particular the concept of a program can be limited to programs that automate previously (or conceivably) manual computer operator activity, which once was part of the field of SNA.[7]

The distinction of hardware and software serves different purposes in different contexts. There is no general rationale for the decomposition of systems in hardware and software which the SNA needs to be aware of. But in particular cases it may be required the responsible SNA investigates the underlying rationale of the hardware software distinction.

**Axiom 3 for SNABOK (snax3, Independence of hardware/software role assignment and evolution):** The rationale for the decomposition of systems in hardware and software varies from context to context. Is is required that an SNA is aware of the rationale (or at least the proposed rationale) of respective roles of hardware and software in a particular setting. Moreover SNA is robust against changes in these roles.

**Axiom 4 for SNABOK (snax4, Software component focus axiom):** Software is composed of components that can be independently acquired in principle. Components have to be installed and configured in order to produce the required functionalities. Adequate configuration of software components is an issue that SNA should consider provided anyone in an organization should have detailed expertise.

**Axiom 5 for SNABOK (snax5, Extreme abstraction view axiom):** SNA takes a more abstract view of computer systems than SWE, as it is not concerned with the technical aspects of control software execution an control software structure. Moreover, SNA abstracts from user intentions to general services and service levels. SNA comprises the highest strategic level of abstraction for the ICT structure of any organization with the possible exception of those aspects that are of top strategic level for an organization.[8]

[7] A formalism for the description of such operator programs can be found in [3].

[8] For instance, if the top strategy dictates to do away with all use of computers and of ICT then there may be no need to involve the local SNAT in that decision. Similarly a decision to go for exclusive use of products of a given

**Axiom 6 for SNABOK (snax6, Computer ignorance axiom):** It is unlikely and unnecessary that SNA's 'know' or understand 'how a computer works'. That kind of knowledge is far too complicated and it is preferable if a SNA has a precise grip on the abstraction levels for which (s)he is in the possession of adequate models. In fact like the understanding of the human brain is often delegated to philosophers, an understanding of the artificial brain should is also difficult task, involving computer science, electronics and quantum physics. It is important that in the setting of SNA the concept of understanding is used with some scientific basis: having available some model which explains fundamental aspects and properties, rather than some form of feeling comfortable with the area on the basis of past experience. in addition a strong inclination is needed to use technical jargon only in the presence of adequate definitions.

# 3 The current state of SNA

It requires a systematic investigation to determine the present state of the art in SNA. Here is some guesswork which may be valid up to an order of magnitude. As a job market the field seems to be local or regional. People expect to talk with their systems group in their local tongue. I tried to make some estimates for the Dutch scene to make an estimation of the amount of teaching and certification that may be needed for SNA in the Netherlands in coming years.

SNA is a functionality present in each independent organization with say 50 or more computers or PC's. As soon as compute servers, file servers, backup services, group licensing and a local area network with its own access policy are in place a systems group with one or more employees, will probably be present. This suggests at least one systems group per 500 workforce, which in its turn suggests the existence of well over a 250.000 SNA groups worldwide, involving perhaps a million workers. In the Netherlands a guess may be: a substantial SNA group with 5 or more members per 10.000 inhabitants at least. This makes the number of SNA personnel comparable to the number of general practitioners, some 7000. If SNA personnel would stay in the field during their entire working career, that would be some 40 years per person, generating a replacement volume of 200 per year. These figures are guesswork in need of further inquiry.

## 3.1 Foreseeable development of SNA

Many different developments have an effect on the position and role of SNA groups. Some of these trends are discussed below.

### 3.1.1 More centralized service centers

SNA is in the process of ongoing concentration with the effect that per workplace fewer SNA personnel are needed. A significant step that may be taken in this respect in the near future is

company may be entirely political to such a degree that involvement of SNA expertise in that decision is adequately judged irrelevant.

that companies as well as non-profit organizations may place the management of their computing facilities in the hands of independent service centers which concentrate processing power in a few locations where highly centralized SNA allows providing the SNA service to many groups of users. High bandwidth connections facilitate the centralization of computing resources better than ever before. Interestingly the development of affordable and increasingly powerful hardware and the massive production of relatively cheap software has created a situation that SNA is so expensive and problematic that the emergence of glass fiber networks may be used to remove PC's and laptops from the working environment. This implies a return to a centralized computing center serving many customers via 'thin' terminals with powerful graphics but no other computing power and preferably without any data storage capacity.

**Axiom 7 for SNABOK (snax7, SNA workload oscillation axiom):** Centralization and de-centralization are oscillating movements dependent on the cumulative effect of a number of technologies. Cost reduction for SNA is a factor that works in the direction of centralization. Presrevation of increase of local autonomy constitutes a force for decentralization of SNA.

### 3.1.2 More security problems because of wireless networking

The requirements on security are ever increasing and the technology poses increasingly formidable complications in relation to security. In particular the ubiquitous use of wireless networking creates a number of additional security worries generating the need for more SNA expertise at least temporarily.

**Axiom 8 for SNABOK (snax8, Security focus axiom):** Taking care of systems security is the most significant complicating factor fro SNA.

### 3.1.3 More systems organization problems because of web dependence

Another trend is the vast increase in the use of the web. Increasingly organizations are under pressure to offer an attractive and consistent website. This turns out to be formidable challenge for each substantial organization. The people needed for resolving these problems need an SNA background rather than an SWE background.

## 4 Boundaries of SNABOK

Just like the SWEBOK, a SNABOK needs to abstract from particular technologies and platforms as far as possible. This will be much harder due to the notoriously dominant position of some platforms and in particular of operating systems. It is open for discussion whether an outline of Tcp/IP and IPv4/IPv6 must be considered specific technologies from which SNABOK should abstract or rather basic principles that SNABOK must convey. I guess that it has to be the latter if SNABOK is to be really useful in the current setting.

## 4.1 What lies outside SNABOK

To begin with most of SWEBOK lies outside SNABOK and so does almost everything related to hardware engineering. In general the competences of the use of systems that are offered to users under the responsibility of a SNAT will exceed those of the members of a SNAT. Typically the content of a website will not be under the control of SNA but the working modes and the organization of the web master team will be. Definitely the tasks that users intend to fulfill by means of the use of offered ICT services will involve many competences and knowledge areas which lie entirely outside the scope of even the most universally minded SNA or SNAT.

## 4.2 Knowledge Areas for SNABOK

Here is a listing of knowledge areas that regard SNA. This listing is preliminary and may change as a consequence of field research and peer review.

1. Access control and security.
2. Device roles and device classification
3. Classification of system behaviors
4. Device composition patterns
5. Software component classification and software component roles
6. Software component configuration
7. Software component acquisition, installation, un-installation,
8. System evolution
9. System architecture design and planning
10. Performance measurement
11. Cost measurement
12. System requirements engineering
13. User support, user satisfaction measurement,
14. People management skills (if SNA is performed by a team)
15. Project management skills

# 5 On devices and device classification

I will elaborate on this knowledge area because it is needed to obtain a workable level of abstraction for a SNABOK. Axioms snax5 and snax6 and to a lesser degree snax2 imply that device classification cannot be based on a technical understanding of device internals or even of device roles. A from scratch and bottom up ontology for computer systems should provided for SNABK. That ontology is separating and partial in that it supports making distinctions between entities based on features and properties while it is not possible with this ontology to obtain full concept definitions needed to allow reliable judgments that an entity instantiates a concept. For that reason the ontology will be considered a partial resolution ontology (PRO) or alternatively a low resolution ontology.

## 5.1 Ontology design and role

The design of ontologies is an area of research in itself. For a comprehensive survey we refer to [6]. Some ad hoc meta-theory of ontologies is presented in order to put the development in a systematic perspective. Besides low resolution ontology, there is room for: medium resolution ontology and high resolution ontology each of them also classified as partial resolution ontology. An ontology may be separating, physical appearance based, technical detail based, role based and process oriented. A (partial resolution) ontology providing a complete definition of one or more concepts provides locally full resolution, and if all of its concepts have proper definitions it is a full resolution ontology (overall).

**Axiom 9 for SNABOK (snax9, Non-circular ontology axiom):** For all kinds of entities, processes and tasks essential for SNA, SNABOK should contain an ontology. Preferably partial resolution ontologies are used with a low degree of resolution.[9] Ontologies should use non-circular feature descriptions. For instance a device classification ontology should minimize criteria about device operation and use. An ontology may be used in various ways and its description should not be cast in terms of a preferred role or usage model.

Because ontologies are a notoriously confusing subject based upon an enormous philosophical literature which can only be understood selectively, I will expand here on the kind of PRO that might work for SNA device classification.

## 5.2 A PRO for device classification in SNABOK

It is argued that some simple SNA tasks can be performed by persons who master an adequate low resolution ontology based on physical device properties. Regarding software a distinction can be made between control software (control data) and data software (content data). Though intuitively appealing this distinction seems not to lead to any useful partial resolution ontology for computer users or for SNA. It sheds some light on an ontology of control software for software engineering,

[9] Thus the focus is on a low level ontology because that is easy to learn and to use by minimizing information content. There is no such thing as 'the relevant BOK' as such about devices for instance, it all relates to the tasks on intends to perform.

including a link with a local full resolution ontology regarding the concepts of computer program and computer programming. For the purpose of SNABOK, however, a process oriented partial resolution software ontology may be developed based on the notion of an active application.

## 5.3 Users of the ontology

The ontology is intended to be of value for people with an interest in the use, maintenance and administration of computer systems and networks. The assumption is that these people may perform their tasks on the basis of a low resolution ontology used to classify relevant entities.

The problem that this ontology design seeks to address is the seeming circularity of most if not all documentation on the subject of computer systems and SNA. Due to a fundamental lack of conceptual definitions most documentation can only be read by those who have an intimate knowledge of the subject already. The acquisition of that knowledge is supposed to take place during teaching of theoretical concepts augmented with practical experience and contacts with people who are supposed to have and usually claim to have intimate subject knowledge. Books on computer systems start using the full jargon from the first page without spending any attention on the different phases of understanding that a reader may be in regarding the terminology that is used.[10]

## 5.4 General properties of PRO's

It may be useful to have some terminology available for PRO's. Here I will discuss: use, soundness, adequacy, task dependence, completeness, maintenance, merging and some open problems.

### 5.4.1 PRO use

A separating ontology provides a number of concepts and concept descriptions that allows one to make classifications without having definitions available. Thus one may classifiy a circular metallic object as a CD or a DVD, without being sure that it contains any optically readable data, just by inspecting its form. If the mouse and the keyboard must be distinguished the use of a partial resolution ontology is a very plausible option. A PRO is used for such questions.[11]

Partial resolution ontology allows one to classify entities over concepts by checking compliance with a number of criteria. Compliance of an entity X with a number of criteria (features, properties)

[10] Readers of technical documentation are mostly expected to be flexible in their understanding of technical terms. For instance if the phrase computer software is used most readers will feel comfortable. They may be unaware that this ease of mind need not rest upon the basis of solid and well-understood definitions but rather on the ability to process a certain brand of terminology. However, if for instance a clear explanation is asked about the difference between computer software and computer firmware some readers may find out that their definitions simply fail to provide the precision required for successfully answering such a question.

[11] However, if one is to decide which of two systems provides adequate anti-virus protection one may be faced with the more fundamental question what is a virus, requiring at least a full resolution island regarding virus infections in one's ontology.

listed for concept Y in ontology Z allows the assertion that 'X is an Y' (understood within the framework Z). One may say that a partial resolution ontology contains partial definitions. If X is a Y in Z one is still faced with the question whether X is really a Y. To answer that question a full resolution ontology is needed and it must be checked that X satisfies all criteria for Y in the full resolution ontology. After classification has been performed the ontology may contain important information about the classified entity. Further if the classification is done as a step in a task the next steps may depend on the result of classification.

### 5.4.2 PRO soundness

Let us assume for the sake of the argument that computer software is said to be a collection of bit sequences physically placed on some form of memory, be it a CD, a hard disk, machine memory of a processor, or a piece of paper. This qualification will not serve as a definition, but it is consistent with the glass fiber definition of proactive software.[12] At the level of abstraction where computer software is said to be a collection of bit sequences placed on some memory, the same holds for firmware and sharper distinctions cannot be drawn. In fact at that level of abstraction all digital data are in the same class. An ontology providing no more information on software than just mentioned may be called a low resolution ontology, at least from the perspective of computer software. It is said to be low resolution because it provides less distinctions and fewer definitions than standard expositions on the subject suggest to be present. The higher the resolution level of an ontology the more distinctions can be made. Because ontologies are always abstractions each ontology is a partial resolution from some perspective of course.[13] Ontologies form hierarchies with respect to their resolution.

If the combined requirements posed on concept X in ontology F cannot reasonably be taken as a definition, F is said to have a partial resolution (at least regarding the concept X). Whether a partial resolution ontology is classified as having low resolution, medium resolution or high resolution is a matter of pragmatic choice in specific circumstances. Moving to higher resolution concepts are equipped with more properties and the classification of entities, as instances of concepts, will lead to fewer false positives. An ontology may be based on a classification of objets according to their physical appearance, or it may use role descriptions. Especially in computing there is much room for ontologies based on properties of (sub)system behavior.

If a low resolution ontology F constitutes the basis of the judgment that entity X is an A, then this may be a false positive. The ontology is called sound if the judgment that X is not an A is always valid. In other words: every A is classified as an A.[14] If X has full resolution w.r.t. the concept A precisely if the converse holds as well.

[12] In saying that a car is an industrial product a car is not defined but nevertheless non-trivial conclusions, for instance that the sun is not a car, can be drawn from that statement.

[13] With the possible exception of elementary physics where it is conceivable that particles are classified with no more than a few mathematical numbers and the theory may exclude the very possibility of further refinement.

[14] Thus it would be acceptable if X is classified as a car, even if its engine has been taken out of it, but it is not acceptable if X fails to be classified as a car due to a missing number plate.

### 5.4.3 PRO adequacy

A PRO can be used as follows: a protocol can be specified for how a person can perform some task. For instance: look at a scene, and place all A's in the area P. Assuming that the person executing this plan is able to decide whether an entity complies with all requirements on an A in the ontology used, and assuming that in the case at hand no false positives occur, the person will be able to complete the job correctly also in the eyes of those using a ontology with higher resolution. Although the person has not been using definitions of A, (s)he could perform well because a sound low level ontology was used in a context where no false positives happened to occur.

Making explicit use of a PRO allows one to have a complete understanding of the use of the jargon at different stages of discourse. This leads to the **adequate resolution ontology axiom**. For any given practical task, including for instance any package of related activities in SNA, an adequate PRO can be found supporting the work. That means that specialists performing tasks in the given area can treat the property combinations of that ontology as if it were concept definitions, because in the setting of their work false positives will not occur. Explanation or proof is not the duty of the practical specialist, but classification is required all the time.

A PRO is adequate for a task in a given operational architecture, that excludes entities that would lead to incorrect classifications, if no more knowledge about any entities is needed than what follows from the properties of objects in the ontology: all required classifications are possible, and all required information is then found with the class descriptions of the ontology.

**Axiom 10 for SNABOK (snax10, Adequate resolution ontology axiom):** For each task there is an adequate PRO with minimal resolution permitting to perform the task with a minimal adequate knowledge level.

### 5.4.4 Task dependence of PRO adequacy

For someone working in a restaurant it may suffice to define beer as the content of a number of predefined bottle types. For a detective investigating a murder case that need not be the most appropriate definition. In [4] L.H.A. Hart describes the problems that arise in matters concerning justice and law when classifications must be made according to an ontology. The issue is that even if a judge intends to follow the law in a formalistic manner difficulties arise if objects have to be classified. Then classification may be difficult in borderline cases. Hart refers to this as the problems of the penumbra. The penumbra covers the many dark conners surrounding the definition of a seemingly clear concept. Is a bottle of beer still a bottle of beer if its contents has been replaced with petrol? If not which changes of content are within the class of bottled beer and when is a step outside that class made? Is it the case that the form of the bottle as well as the announcement made on its exterior make one more readily inclined to speak about a bottle of beer (independently of the actual contents) than if correct content (beer) were to be presented misleadingly in a wine bottle. Undoubtedly these problems emerge in the setting of computers systems tasks just as well. For each plan that a maintenance engineer has to perform classification questions have to be settled. When this becomes difficult, for instance due to unnoticed changes in technology, the engineer may need to find out what the criterion ought to be instead. But this may lead to faults. Clearly the

optimal office setting will be designed in such a way as to minimize the problems of the penumbra. If, however, problems of the penumbra occur very frequently it may be useful to move from, say a medium resolution ontology based on physical appearance, to either a higher resolution ontology or to a role based ontology, if reasoning from observed roles is simpler.

Ultimately there may be no escape from local full resolution ontologies, though problems of the penumbra cannot be excluded. The only advantage is that the engineer has more possibilities to resolve the classification difficulties according to the intentions of the plan that is to be carried out. Hart uses his focus on the problems of the penumbra to illustrate that in law a purely formalistic approach cannot exist because logical formalism cannot resolve all classification problems concerning particulars. This also holds in the overall setting of computer technology, but it may not hold in a much more restricted setting where the general design has been done with the explicit purpose to make all classification of particular objects simple and reliable on the basis of a low resolution ontology.

### 5.4.5 PRO completeness

A full resolution ontology (FRO) provides a definition for each of its concepts. A definition allows (in principle) definitive judgments of the form X is an A. This is due to the fact that a definition provides necessary and sufficient criteria for membership of the concept A. An FRO is just a complete PRO.

Full resolution ontologies are very hard to obtain. Several problems occur: it is an enormous task to provide full resolution ontologies for all but the simplest settings, and more importantly people differ largely on what they consider adequate definitions of concepts. Therefore it is far more likely that two persons agree on a low or medium resolution ontology than that agreement is found regarding full resolution ontologies. Full resolution will often involve the use of mathematical models, which by itself may be a barrier for understanding even more problematic than the absence of definitions.

Instead of aiming for full resolution ontologies for an entire subject area, one may prefer to develop an ontology that is locally full resolution concerning a limited number of concepts. The concepts are full resolution islands within a partial resolution ontology.

In mathematics people have achieved enormous agreement on full resolution ontologies. In physics that is to some extent the same. In computing full resolution ontologies have been designed to general satisfaction for a part of the mathematical theory of computation, whereas the design of full resolution ontologies for programming and systems construction has proven very difficult.

### 5.4.6 PRO merging

Given two ontologies it is possible in principle to combine them to a joint ontology by classifying X as an A if and only if both ontologies lead to that classification. It is conceivable that two ontologies P and Q are inconsistent. That happens if some important classes are left without any inhabitants.

It is conceivable that P requires a mouse to have a connection wire and that Q requires a mouse to have at least two buttons. Now a wireless single button mouse is falsely not classified as a mouse in the combined ontology. It follows that when two ontologies are combined each category must be checked for consistency of the imported requirements. If an inconsistency arises at least one of the ontologies must be made more liberal by admitting entities in a class it previously did not.

Different full resolution ontologies are necessarily inconsistent and need to be modified rather than made more liberal if an agreement is to be found. That by itself constitutes a reason not to strive for full resolution ontologies for supporting the tasks of SNA.

### 5.4.7 PRO maintenance

A remarkable disadvantage of physical appearance based partial resolution ontologies is that false negatives may result from aging. Some years ago a telephone was a device with a wire connecting it to a wall socket. The wall socket then stood for the identity of the phone in terms of: where is a call going to. Recently phones have become wireless, making it significantly harder to locate the place where a call can be detected. Now an ontology requiring a phone to have a hardwired connection to a wall socket is outdated and it may lead to false negatives. This may be taken to mean that technical knowledge quickly outdates, but that is not so. What happens is that a default attribute (fixed line) moves from a default status tot a non-default status.

Additional categories with novel values for default attributes may be introduced as a part of maintenance. Besides the wireless phone there is now also the mobile phone. For some reason mobile phones are always wireless, whereas it is simple to imagine that one uses a mobile device which may be plugged into a wall socket and then behaves like a non-moving mobile phone but one without batteries, with high data communication capacity, without any worries about base station coverage and also without any health related uncertainties. The use of such systems might even be cheaper than current mobile phone technology.[15]

Now the use of architectures may prolong the life time of an ontology because the place of an entity in an architecture may be considered a role and roles may easily survive technology dependent physical equipment details.

### 5.4.8 Open problems PRO resolution for SNABOK

Here are some questions that require further investigation:

- It is necessary for any aspect or part of SNABOK to employ an ontology for software that makes a distinction between control software and data software.
- Is there any need in SWABOK to include locally full resolution ontologies for any concept?

[15] This is automatic call forwarding setup via device plugin.

- Is it characteristic for SNA as an administration discipline that it has no need of concept definitions (i.e. locally full resolution ontologies), in contrast to an engineering discipline where a full resolution ontology may be preferred even if false negatives result for the simple reason engineering is supposed to produce category instances for an external world.

# 6 Roles: roles may survive features

The device allowing one to perform an outgoing phone call has a remarkable role stability. If one intends to phone some number, all devices are equally workable, though there may be differences in price and quality. Let us call this an outgoing phone call device. That device may be simply described in terms of the role it plays in supporting outgoing calls. Now once that role has been described, it is possible to forget about the specific technologies used to support it and an outgoing phone call device becomes any device capable of playing that role.

This suggests that a role based ontology can be more stable than a physical appearance based one because appearances change much more quickly than roles. Unfortunately roles are hard to define in a non-circular fashion. If one intends to use a mobile laptop, for security reasons some initialization steps involving passwords and authentication may be performed by necessity via a fixed connection network, less amenable to security risks. This indicates that the concept of a wired connection is still very important but now its use has been changed from the connectivity per se to the administrative layer supporting mobile connectivity.

A role definition may arise as follows: given device (device type) $D$ and a form of its use $U$, then define $U$-ing as using functionality $U$ of a $D$. Now two roles emerge: being a $U$-er and being a $U$-tool. The $U$-er is a person who performs $U$ but forgetting all technical details and focussing exclusively on what the person needs to achieve. A $U$-tool is any appropriate tool for facilitating $U$-ing.

## 6.1 Original role persistency

It is useful to take the development of technology explicitly into consideration when working towards new ontologies. The reason for this is that roles are derived from objectives and that objectives are often present in early stages of a technology already. Thus for someone interested in the development of an ontology for mobile phones it is definitely useful to define the fixed line connection ontology first. That makes it far easier to see which devices are role successors of other devices. Thus, although technology may change and role players may change their technology, this only adds to the number of ways in which a role can be played and old ways to do so become restricted to more specific cases, in fact often cases that require a more than average or minimal quality of service.

## 6.2 The ontology process

In SWE the software process relates to all phases of the software life-cycle. For the development of ontologies there will be a need for an explicitly managed ontology process jus as well. A possible creation path (ontology process) of an ontology for devices as a part of SNABOK may be presented in some detail as follows:

1. An initial PRO (I-PRO) based on the phyiscal apperance of entities is needed to start any discussion of the subject. In the absence of such an I-PRO one may have a thorough mathematical and logical understanding of computing but lack the ability to recognize and properly use computing devices. Pictures and physical demonstrations are a most useful way to explain an I-PRO.

2. A more refined physical detail PRO (PD-PRO) is developed subsequently. The number of pins on a connector and the voltages potentally present on these pins may be part of a PD-PRO entry for some component. Here are some other examples: the numbers of pixels on a screen, the size of a monitor, the number of colors available on a monitor, the electrical properties of a connecting cable, its length and color, the stated capacity of a connection, the precise lay out of a key board, the number of components in a crate, the number of ports of a switch.

3. For today's computing a DP-PRO allowing for some 50 kinds of equipment probably provides a plausible basis for a computer expert. Here is a listing of component names that might occur as concepts in an I-PRO, (to be refined in a PD-PRO). The grouping used is according to the degree of immediate visibility of the equipment for a hypothetical end user.

   - wired/wireless mouse, mouse pad, wired/wireless key board, CRT monitor, flatscreen, VGA monitor, VGA/USB adapter, PC, workstation, laptop, notebook, anti-theft cable, battery unit, digital camera, webcam, microphone, speaker, floppy disk, CD, DVD, CD writer, ethernet wall socket, ethernet cable, ethernet plug, USB hub, USB cable, USB plug, Fire Wire plug, Fire Wire cable, power cable, power adapter, airport card, video card, (color) photo copier, fax, (color) printer, copier/fax/printer paper, toner cartridge, scanner, external disk unit, beamer, electronic wall-screen.
   - telephone modem, video cable modem, ADSL modem, cable modem, phone cable, phone wall socket, base station, router, fixed telephone, mobile phone, telephone plug.
   - server, original Linux[16] box, original M-OS9 box, original OS X box, original Unix BSD box, original W98 box, original NT4 box, original XP box..., switch, wall socket, crate,
   - UPS (uninterruptable power supply unit), ventilation unit, security lock, fire protection equipment, tape robot, backup disk unit.

[16] Box is an alternative for machine. There is no subject oriented classification of operating systems available which forces one to use brand names and to worry about trademarks at a point that this is totally irrelevant in principle. This information can only be taken from the documentation of machines with pre-installed operating systems. This means that one needs to read from the documentation that a particular laptop is in fact an OS X box and so on. Lacking such information a full classification cannot be provided.

Notice that this condition implies that one need not know in any way what an operating system is nor how to switch on a machine to see what it has to say about itself. It forces one to use the prefix original for the box qualifications if it turns out that during its life a P box may become a Q box and so on. If such transitions can take place, it is impossible to determine that it is a Q box by simply inspecting the original documentation.

The above listing of devices is surprisingly simple and the main explanation for the remarkable complexity of today's computing systems from the user perspective lies in the difficulties of software classification.

4. A PD-PRO allowing to determine more technical information about physical devices occurring in the I-PRO may be given by adding no more than say ten information fields to most concepts. When making a classification of an item (or device) not only the name of concept it instantiates must be determined but also various information fields providing detail specification information need to be instantiated. In most cases the information involved needs to be taken from the documentation available for concept instances (i.e. devices).

   Refining to a DP-PRO may generate many time consuming puzzles. Consider for instance the case of the mouse. An attempt to describe the mouse in words may read as follows: to begin with a mouse is a device. Its size is such that it easily fits in a human hand. Its weight allows being moved by hand easily over a flat surface over distances up to say 20 centimeters. Moving it in a controlled fashion for a few millimeters must be possible too. It is wired to the system, e.g. with a USB cable. It has between 1 and four buttons. It can be used (moved and have its buttons pushed) with one hand. It is operated on a surface that has some surface profile, either physically or (for an optical mouse) in its color pattern. The mouse has a sensor allowing it to track its movements and the sensor data are continuously communicated to the system. This sensor is at the bottom of the device. A wireless mouse operates without the mentioned wire, while communicating with 'the system' by means of a wireless communication protocol, e.g. Bluetooth. Now if there are three wireless mouses around it may become difficult to identify 'the mouse' in some setting. Therefore it is a design rule for a computing context to have at most one mouse around, per installation. However, for a maintenance engineer having to replace the mouse this may be non-obvious as (s)he may need to engage in a setting where a second mouse enters the scene. Already at this stage a maintenance engineer needs to have a dynamic view on system architectures, in order to be able to read and execute in a reliable fashion his/her plan of action.

   As it emerges from the issue with the wire, the kind of description above is quite technology dependent. For that reason it is useful to illustrate the low resolution ontology with images of the objects discussed. These images provide much additional information, making the user quickly aware of a technology change that may have taken place and of a need to reconsider the ontology before moving ahead. Indeed a low resolution ontology based on physical appearance should be provided as a sequence of historically ordered ontologies, each of them provided with additional data such as images, greatly simplifying the classification efforts, and simultaneously making the ontology user aware of both the evolution of technology through time and of unexpected technology changes requiring extensions or modifications of the ontology. Having the temporal development of an ontology available greatly simplifies thinking in terms of roles.

5. An initial role PD-PRO (IR-PD-PRO) refines a PD-PRO by explaining for all entities the roles they may play in system processes. The processes themselves will also appear as entities in an IR-PD-PRO.

6. A local full resolution ontology (L-FRO extending IR-PD-PRO) may provide complete definitions for a subset of concepts. For some concepts, like a system or a network a L-FRO will provide no additional information as these terms represent concepts which are used at quite high level of abstraction only.

7. L-FRO's constitute models of a concept or concept family. There may be many competing L-FCO's that provide a full resolution ontology for a given concept. These may be viewed as competing definitions.

**Axiom 11 for SNABOK (snax11, Roles after features axiom):** Role oriented ontologies are based on feature based ontologies. Role based ontologies may provide better heuristics for classification but are weaker when classification problems arise if roles cannot be determined.

### 6.3 PRO based competences

Here we focus in particular on the competences which may be acquired by a person having no more subject knowledge available than the ability to make device classifications classifications according to a given PD-PRO. The underlying assumption states that it is very well feasible to make accurate classifications pieces of equipment without having (or having ever had) any information or understanding concerning the role or purpose of such equipment. In order to distinguish a mouse from a keyboard one need not know or understand the purpose of these devices, let alone how their cooperation may be organized. It is assumed that device documentation may be used for the classification exercise. Thus if a piece of equipment is described as a router and it is a box with a power connection, and a number of other connectors, not provided with a screen, one may, in orderly circumstances, safely assume that it is a router indeed.

### 6.4 PRO based competence examples

Here are some tasks that may be carried out on by a person or a team of persons mastering a reasonably sophisticated PD-PRO for computing equipment.[17]

1. Given a number of rooms provided with adequate cable connections and a drawing of how to setup an equipment installation, as well as the right stock of equipment put everything in place and connect all relevant wires.
2. Given a space with equipment it is very likely that a person can be instructed to replace a (potentially flawed) device by a newer one (given the new one). The point is that the person needs no more geographical coordinates than whatever expert would need, being able to use his/her classification ability to spot concept instances in a limited area.[18]
3. When a postal service delivers a package with a piece of equipment the classification ability can be used to sign for accepting the item as being an instance of the said concept indeed.

[17] For SNABOK it should take no more than some 250 pages to document the PD-PRO and it should be a matter of weeks to familiarize a wholly uninitiated person with its content to the point that, with the document as a help reliable item classifications can be performed.

[18] The task to test the correct working of the replaced item is far beyond this competence, and equally so is the task to find out which one of a given limited number of concept instances may be defective.

4. To buy a device in a shop, equipped with no more than a description of its PD-PCO classification and some cash. If the shop offers many variants of the device differing in parameters lacking from the description at hand the person will have no clue on to how to proceed, however.

In the strategy proposed here the first stage of understanding the vocabulary involved is to view it as an ontology where a global indication of the kind of object is presented based on physical appearance and properties, and it is suggested that task which can be performed with only this amount of knowledge should be carried out that way without hesitation.[19] E.g. a CD is a metallic disk of a diameter of some 10 centimeters and about 2 millimeter thick. A person sent to buy a CD needs no more information than this. No insight in CD technology or use is required for it.

So whether or not these descriptions qualify as definitions of concepts, they certainly allow one to separate categories, which is what matters for simple tasks because it is hardly imaginable that one may be in doubt whether an entitity is a CD or a key board. Summarizing an PD-PRO and ID-PD-PRO, can help a person to make plausible decisions in many circumstances and to make reliable decisions in specifically well-organized circumstances.

### 6.4.1 Preparing the scene for low resolution ontology based competences

For a low resolution ontology based competence to be effective in a setting it is essential that the number of classification errors is minimal. There are two ways to prevent classification errors: (i) to use personnel who have a far higher resolution ontology at their disposal, taking recourse to that in the case of any doubt, (ii) to organize the setting in such a way that a certain low resolution based competence produces no false positives and permits the classification of nearly every item that may occur, with a guide on what to do if the classification effort fails (an exception).

Obviously the second strategy is superior in the sense that it allows to minimize the amount of knowledge (the theoretical basis of the classification competence) needed to perform a range of tasks. This has two advantages: the knowledge can be taught much more quickly to anybody potentially involved in the work processes and the knowledge can be made independent from other bodies of knowledge making it much easier to maintain. In the era of the so-called knowledge economy one must be economic with knowledge, which implies that work processes must be organized in such a way that tasks can be performed with a minimum of knowledge[20]

In an office where a mouse looks like a keyboard, a phone is built in in the mouse, the routers are placed in the table and so on, naive workers having a low resolution ontology at their disposal will be ill-prepared. A clear symbolic organization of the setting geared towards the effectiveness of a shared low resolution ontology is therefore very practical. In the military profession one can find

[19] In the medical profession there is a common argument against this: if help treat a patient with some problem it is far better if you are able to detect related problems from the perspective of a broader knowledge base. In SNA that argument would be overdone. The decision to perform a certain action should be taken with care of course, but its execution may be done according to the book.

[20] The widely advocated opinion that education in science and technology is mandatory for survival in the knowledge economy age cannot be taken for granted at all, unless a systematic underpinning is given by means of a detailed knowledge management analysis of a vast number of work processes.

out who is in charge by simply looking at someone's physical appearance. An easily taught PD-PRO provides full information about the power structure. This is not at all the case in a modern office organization.

# 7 Ontology for system operation

When one intends to work with a computer system there is no way around the use of an ontology for items that escape simple physical observation. In the case of human beings everyone is used to the concept of a person performing an action without having a clue as to the physical basis of that view as an electrochemical phenomenon inside the person's body. This very similarity should be taken as a point of departure for the understanding of computer systems operation as well. Clearly the chances of detecting the physical basis of observed phenomena is an order of magnitude better than in the case of human behavior. But the chance of acquiring an effective, consistent and simple mental model is low as well.

For this reason it is a reasonable idea to forget about the inner structure of a system or network or its components when focussing on its operation. The nearest thing to a DP-PRO is a MTVE-PRO: a multi-thread visible event PRO. A thread is a (hypothetical) succession of events grouped by 'being in the same thread'. This grouping is admittedly artificial. Nevertheless by recording the visible details of events (e.g. printed pages, monitor screens, incoming emails, bleeps and other sounds) a collection of events may be generated. A DP-PRO may be used to classify these over various threads (i.e. as belonging together or not). Then an MTVE-PRO may produce judgments of the form: I am now producing an email on window 3, I am text editing in window 4, I am monitoring the execution of task P via window 2.

Another example: I have surfed to the website of my institution (from home), then I tried to login of service S which was refused to me because of an inadequate passphrase, upon which I have mailed a request to "support" (the SNAT of my employer). I got a reply of my friend F in SNAT. Not being a professional SNA he had no clue and suggested me to choose a new password, which is impossible due to the very problem that I have experienced. I have phoned F and he told me that this problem requires me to show my face on site.

The simplicity of MTV-PRO is that it refrains from the identification of so-called services. A question like: 'are we offering service P compliant with the service level agreement that we have signed' lies outside the range of MTVE-PRO. The demonstration of a counter example, however, may lie with in the scope of a competence supported by an adequate MTVE-PRO.

## 7.1 System operation outside any simple MTVE-PRO

System operation refers to all operational aspects that may be relevant for SNA. Some aspects are definitely outside the scope of any simple and user oriented MTVE-PRO. Here is a question that SNAT may have to answer for its manager: which are the security implications of the fact that the current internet technology uses packet switching technology?

It is difficult to determine at which level of abstraction this question should be accessed. If it involves a full understanding of all equipment and protocols involved that will escape the abilities of nearly every user, and in particular of the vast majority of users who have serious security concerns generated by their activities outside computing. Other questions that are difficult to answer as well: what is an email service, what is a document, what is open source licensing, what is an executable file, what is a web-site, what is a browser. Initially the definition seems to be easy by giving examples and stating objectives. Significant difficulties arise then if external concepts like interoperability, standard compliance, web-enabledness, comprehensibility, and temporal persistence enter the discussion. All of a sudden a certain form of interoperability may be considered crucial for the definition of an email service thereby enforcing the change of viewpoint that a service that was considered an email service is now considered an email look-alike only, and is for that reason in urgent need for replacement.

**Axiom 12 of SNABOK (snax12, full resolution for threads avoidance axiom):** An operational ontology for SNABOK need not exceed the qualities of an MTVE-PRO. Higher resolution ontologies and their use are the domain of system engineers, including software engineers. Importing an ontology for processes and threads should be avoided.

# 8 SNABOK milestone planning

The first milestone is achieved if a number of missing link documents have been written and local field studies have been carried out. Then a consortium should be formed that is able to produce a Stoneman version that may count on some approval in the field. From that point onwards the project planning of the SWEBOK effort is a perfect example of how this kind of endeavor may be carried out effectively. At this stage I only describe what I will try to achieve with the local field studies in the Netherlands in the form of a questionnaire that will be used for interviews. During the field studies the questionnaire may be adapted as well as the MLDF which may evolve through different versions. More information is needed on the background and experience of the current SNA workforce. This information must be retrieved via field studies. Equally unknown is to what extent SNA professionals are lacking competences and missing a common language as well as an approved common body of knowledge. Indeed the unit (subject) of a field study is a SNAT, where it is taken for granted that, by definition a SNAT contains some SNA's. Questions are preferably discussed with a leading person in the team who is willing to consider him/herself a SNA. This sample questionnaire covers the kind of information I would like to collect.

1. Do you agree that you are active as an SNA (i.e. that is an important role that you play) working inside a SNAT? Which terms do you use for the whole team, for its subteams and the different roles that you distinguish within the team and its subteams.
2. How many people work in this SNAT? What is their educational background? How many users is it working for? How many PC's, laptops, servers, routers and mainframes are being administered? How do you measure the size of your operation? What kind of measurement or performance estimate justifies the size of the team.
3. Do you agree that an important part of the knowledge used by the members in this SNAT may be represented at an abstraction level which ignores the particular platforms, hardware

products and software components that your SNAT is dealing with. If so: can you give some exposition on what these more abstract issues are. If not: is your perspective of SNA moving into a direction where that is the case, or is it just the other way around.

4. Do you consider it relevant that a SNA has a technical awareness of SWE, and if so do you hold as well that the SNA needs an awareness of computer hardware engineering and electro-technical engineering. Given the abstraction level of SWE which will not dig into the details of any special program notation or programming environment, is this awareness of SWE sufficient or are more specific SWE skills required not explicitly covered by SWEBOK?
5. If a SNABOK were available with some international status: would you use it?
6. To what extent will your work and that of your SNAT be automated in the coming years, and what impact on the SNAT is that likely to have?
7. Are you involved in the strategic decision of the organization that you serve in a sufficient way, is this involvement sufficient given a long term ambition to remain operational in the area of SNA?
8. Do you make a distinction between knowledge and competences? If so can you give clarifying examples? Which knowledge and which competences are missing from some or all members of your team? How is this lack of knowledge or lack of competence resolved?
9. Does it happen that members of your SNAT know to much, i.e. that due to having more background knowledge than required they fail to work at the required abstraction level, thus rendering their operation less effective than expected? Do you use or need an explicit method for unlearning obsolete or superfluous knowledge or competence?
10. Can you provide a survey of the kinds of equipment that you are using, together with a description of their roles?
11. Can you give definitions of the following concepts (decreasing in relevance): operating system, computer, computer software, user, system administrator, computer virus, computer worm.

## References


[1] J.A. Bergstra. Machine function based control code algebras. In M.Bonsangue and F.de Boer, editors, *proceedings FCMO2003, to appear*, page ?? Springer Verlag, 2004.

[2] J.A. Bergstra and S.F.M. van Vlijmen. *Theoretische Software-Engineering*. ZENO-Institute, Leiden, Utrecht, The Netherlands, 1998. In Dutch.

[3] Jan Bergstra and Pum Walters. Operator programs and operator processes. *Information and Software Technology*, 45:681–689, 2003.

[4] H.L.A.Hart. Positivism and the separation of law and morals. In R.M.Dworkin, editor, *The Philosophy of law*, pages 17–38. Oxford University Press, 1977.

[5] J.Earley and H.Sturgis. A formalism for translator interactions. *CACM*, 13(10):607–617, 1970.

[6] Roberto Poli. Alwis: ontology for knowledge engineers. In *Questiones Infinitae*. Zeno, the Leiden Utrecht Institute of Philosophy, 2001.