Is it $\phi\tau\psi$ or bullshit?

Afscheidsrede 16 oktober 2009

Prof.Dr.ir. Jan L.G. Dietz

TU Delft
Technische Universiteit Delft

Faculteit Elektrotechniek, Wiskunde en Informatica

Is it $\phi\tau\psi$ or bullshit?

Tekst bij de afscheidsrede
uitgesproken op 16 oktober 2009
door

Prof.Dr.ir. Jan L.G. Dietz
Hoogleraar Ontwerpen van Informatiesystemen

Foto omslag: Ingrid Brinkhuis en Yasmine Vo

"Success is like a liberation, or the first phase of a love affair"
(Jeanne Moreau)

Mijnheer de Rector Magnificus,
leden van het College van Bestuur.
Collegae hoogleraren en andere leden
van de universitaire gemeenschap.
Zeer gewaardeerde toehoorders.
Dames en heren.

A farewell lecture is the opportunity par excellence to reflect on one's academic career, and to extend that reflection to one's total professional life, as well as to one's personal life. It is also the last chance to state ex cathedra one's main certainties and concerns about the state of the discipline one has pursued.

The reader will have noticed that I changed language. I switched from Nederlands to WoLa (which stands for "World Language"). WoLa is the international language that is emerging from US English, as the natural alternative to Esperanto, the language that was constructed at the end of the 19th century and that has still some 2 million supporters. I expect that in two generations half of the world population is speaking WoLa. By that time there will be a governing board, operating over the internet, that determines the rules of grammar and of pronunciation, such that people from all mother tongues are able to speak it. In three generations, the first reviewer of a scientific paper will comment: "I advise the authors to have the text being corrected by a native WoLa speaking person".

The current practice of improving the operation of enterprises, in particular by applying Information and Communication Technology (ICT) can confidently be called disastrous. Despite the availa-

bility of appropriate knowledge, practitioners (who usually call themselves information architect, business architect, or enterprise architect) stick to so-called best practices, which lack any scientific justification. Theoretically solid knowledge and practically useful methodologies based on it, are brushed aside for allegedly being too complicated or 'not invented here'. Let me draw the analogy with an engineering discipline that is only some 50 years older, namely aeronautical engineering, to illustrate how disastrous the current situation is. Imagine that aeronautical engineers, in designing and building an aircraft, would ignore the laws of aerodynamics for being too complicated, and would refuse to consider demonstrably better materials and building techniques for being 'not invented here'. Obviously, such a practice would immediately lead to broad societal protests and to the fall of governments. Not so for the engineering discipline of applying ICT in enterprises. There are no societal protests and governments do not fall. Apparently, society accepts the frequent malfunctioning of vital systems, the irritating non-compliancy of ICT artifacts with human needs and expectations, and the continuous huge waste of money.

It is this disastrous current situation that has lead me to coin the title "Is it $\phi\tau\psi$ or bullshit?" for my farewell lecture. The word that is formed by the Greek letters ϕ, τ, and ψ refers to what is theoretically sound in general. More specifically, the letters refer to three coherent theories that I have partly created and partly composed of existing ones. As for the word "bullshit", this may sound harsh and vulgar, but it need not be. Bullshit is the subject of a serious philosophical study by Harry Frankfurt[1]. Let me quote from his book:

"Bullshit is unavoidable whenever circumstances require someone to talk without knowing what he is talking about."

[1] Frankfurt, H.G.: *On Bullshit*, Princeton University Press, New Jersey, 2005

Alas, such circumstances are omnipresent in modern society. Typical bullshitters are consultants (of all kinds), since it is of vital importance for them to appear knowledgeable in the eyes of the client. Unfortunately, many practitioners and researchers in the discipline I am discussing, also belong to the class of bullshitters. In addition, most of them seem to be unaware of their incompetence. This may look like an excuse, but actually it makes things worse, as Frankfurt teaches us:

"Bullshitters retreat from searching for correctness to personal sincerity. Since they cannot be true to the facts, they try to be true to themselves. This is the core of bullshit".

The last quote fits perfectly well my experiences in discussing the state of the art in our field with contemporary practitioners as well as academics over the past decades. Many of them really believe that it is sufficient to be honest to yourself and to others, 'to do your best', as I often hear them defend their professional malperformance.

It is rather naive to think that bullshit can ever be exterminated. Somehow, people seem to need it, even to like it, judging by its luxuriant flowering all over the place. But, it can and must be banished from any serious profession, like ours. Seeking for personal sincerity instead of objective truth and correctness in research and education, or in the practice of advising enterprises and developing ICT artifacts, is no less than a societal crime, I think.

Ouverture

It was in the fall of 1968 that I took the elective course "The art of programming" from Prof. Edsger Dijkstra († 2002), who was freshly appointed at the TU Eindhoven, where I studied Electrical Engineering. Never before had I been so excited and so inspired by lectures. It was soon clear that this was what I wanted to do: programming digital computers. In the summer of the same year I completed a two-month internship at Philips Laboratories in Paris, during which I developed a simulation model for studying the control of traffic in computer networks. I had to program the model in Fortran. It caused me a lot of trouble although at the time I did not understand why. After Dijkstra's lectures, I did; I had learned the important difference between constructing an abstract machine (a mathematical automaton) that would solve a problem and the implementation of that machine in a suitable programming language. I learned that designing software can only become intellectually manageable if these three techniques are well understood and applied:

1. separation of concerns
2. effective use of abstraction
3. devising appropriate concepts

Later I understood that these intellectual techniques apply to the design of all kinds of systems, including information systems and organizations.

Although my specialization was Control Engineering, my responsible professor, Pieter Eyckhoff († 2000), allowed me to do the final thesis work in the faculty of Industrial Engineering. My supervisor over there, Joep Kerbosch, wanted to have IBM's discrete simulation package GPSS (General Purpose Simulation System) be re-engineered and implemented in BEA (Burroughs Extended Algol), a variant of Algol 60, such that one could have full insight

in the actual event processing and statistical analyses, without compromising the completeness of GPSS and the attractive flow-like way of expressing simulation models. I finished the work in six months, delivering a simulation package that consisted of about 40 BEA procedures that collectively did the job. Running the package consumed less than 5% of the resources that GPSS needed, and it was free of bugs. ALgpss, as I had named it, was a jewel; I was proud of it.

Right after my graduation, in 1970, I started my professional career at Philips' Machine Factories in Eindhoven, in the department of Information Systems & Automation. The first thing I had to do was to take a 10-weeks course in COBOL (sic!), followed by an 8-weeks course in file organization. I worked as programmer, system analyst, information analyst, database designer, and project leader. I developed a number of applications, in production control and inventory control, but I was not really proud of them; they were not jewels. After 5 years I found myself in the position of information manager of two factories: it was time to look for new challenges. Fortunately, I met the director of the computing center of the TU Eindhoven at the bar after a meeting of the NRMG (Nederlands RekenMachine Genootschap). Ben Morselt was his name. He offered me to become the scientific liaison officer for the faculty of Industrial Engineering for half of the time and deputy director for the other half. I took the job. What he wanted me to do for the deputy director part, was to relieve him from the burden of distributing the computer resources of the Burroughs B6700 main frame (later replaced by a B7700) in a fair way amongst its users, and to be the financial authority in the negotiations with computer suppliers.

Regarding the fair distribution of computer resources, I introduced 'computing money', with the currency CU (computing unit). For CU's one could buy computer resources. In 1976 there were 5000 users; the annual increase was about 2000 and the annual decrease about 1000. Every organizational unit and every person

could get a so-called user number, a unique numerical code. In order to actually use computer resources, one had to find a sponsor, i.e. another user who was willing to provide CU's. To that end, the sponsor was able to create (directly on the Burroughs main frame) a computer account of the form U<user number of sponsoree>S<user number of sponsor>. Typically, organizational units played the role of sponsor. It was automatically guarded that the collection of accounts constituted a pure tree structure, except for the leaves. The only thing I had to do was to deposit every month an amount of money on the 'top' account of every faculty, conform the decision of the University Computer Board (of which I was the secretary). The internal distribution was the responsibility of the faculty. The Computer Accounting System (CAS) that I developed comprised about 10 pages programming code in BEA (free of bugs), in which heavy use was made of recursion, an incredibly elegant and powerful programming technique, running incredibly fast on the Burroughs main frames since they were stack computers. Burroughs main frames, by the way, are the finest main frames I have ever seen. Only 0.2 fte administrative assistance was needed, whereas comparable computer centers used 3 to 4 fte. CAS was another jewel, and I was proud of it.

Early 1977 my friend Jan Hurstjes (who was student assistant with me at the time) and I conceived the plan to build a Theatre Reservation System for the city theatre in Eindhoven, in our private time. He also brought in a fellow student, Jan van Hoek. We coined the name AURA for the system. For the programming, Jan and Jan used the package WORKER, that they had constructed in cooperation with other students (and staff members). WORKER was a very sophisticated software engineering toolkit. AURA was operational in August of the same year. It run on the Burroughs main frame, and it operated via 8 fixed telephone connections between the city theatre and the computing center. AURA was another jewel, and we were proud of it. No other city theatre in the Netherlands had such an advanced reservation system.

In 1980 I got a full academic position in the group Management Information Systems & Automation, headed by Theo Bemelmans, at the TU Eindhoven, in the faculty of Industrial Engineering. I did research and lectured in information system development methodologies for many years. In 1985 I started to write my dissertation on the formal specification of information systems, supervised by Theo Bemelmans and Kees van Hee (from the faculty of Mathematics and Computer Science). In 1987, Theo encouraged me to apply for a new professorship in Management Information Systems at the University of Maastricht, in the faculty of Economics and Business Administration. I did and I got it, however under the condition that I would start on 1 January 1988 and that I had my doctoral degree by that time. I forced myself to finalize the dissertation in time. Once in Maastricht, I came into contact with Guy Widdershoven, a philosopher at the faculty of Health, Medicine and Life Sciences, who brought Language Philosophy and Social Action Theory to my attention. It was a major missing link in my search for modeling the essence of organizations. In 1992, while looking through the window of my hotel room to the beautiful Mount Fuji, I assigned the name "DEMO" to the modeling approach I was developing (reading: Dynamic Essential Modeling), in order to demonstrate that well-founded methodologies are possible.

Soon after my start in Maastricht I joined the inter faculty cooperative of 'computer science' colleagues, consisting of Hans Crombag and Jaap Hage (faculty of Law), Arie Hasman (faculty of Health, Medicine and Life Sciences) and Jaap van den Herik (faculty of Mathematics and Computer Science). This informal cooperative turned out to be of vital importance. For many years we supervised a common pool of doctoral students, and discussed many other issues. In 1993 I discovered that the research policy of my own faculty had become an obstacle for pursuing the mission that I would later call enterprise engineering. I wanted to go back to a university of technology. "But, what I had learned from the economists about organizations would appear invaluable".

In 1994 I got the chair of Information Systems Design at the TU Delft, in the faculty of Technical Mathematics and Computer Science. It had become available after my predecessor, Henk Sol, had moved to the faculty of Technology, Policy and Management. I could continue pursuing the mission of Enterprise engineering. That was not easy, however. A breakthrough took place in 2004, when the master program "Information Architecture" started. It was a collaboration of my own faculty and the faculty of Technology, Policy and Management (TPM), in particular my chair and the one of René Wagenaar, who passed away in early 2007, far too young. This master program allowed me to introduce my ideas of enterprise engineering. Since René Wagenaar was the only cornerstone in TPM, his death also meant a stop for the combined development of enterprise engineering by the two faculties.

Since September 2008, I am professor catedrático convidado (for 0%) at the TU Lisboa, in the institute IST, and consequently member of the group "DEI professores". Having had a classical education, I was very pleased by that name. It was somewhat disappointing to discover later that "DEI" stood for "Departamento de Engenharia Informaticá".

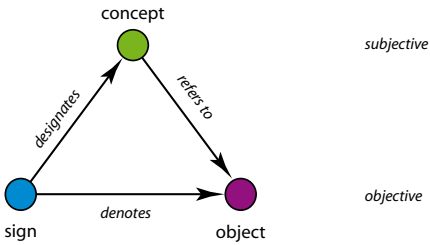
Narrante con espressione

Since the 60's of the 20th century people try to apply ICT in enterprises, for the benefit of all stakeholders. Initially, this field of human endeavor was called Electronic Data Processing (EDP). Its pursuit was to mimic the way human beings handle data by means of computers. In the 70's, the focus shifted from the form of data to their content. EDP became Information Processing, performed by (automated) Information Systems (IS). It meant a paradigm shift for the IS professionals. Now the pursuit was to mimic the cognitive abilities of human beings. But what does the new paradigm exactly look like? What is the difference between data and information? What is a fact? What is an entity?

Below, the ϕ-theory is summarized (the Greek letter "ϕ" is pronounced as "FI", which are the first letters of "Fact" and "Information"). It provides full and deep answers to these key questions, and many more. It is a theory about the conceptualization of factual knowledge. The ϕ-theory is rooted in semiotics, in ontology (including mereology), and in logic.

The semiotic triangle

The *semiotic triangle* portrays the important differentiation between a thought (concept) that is in the mind of a subject (i.e. a human being), the word or symbol (sign) that designates the concept and that he or she uses to communicate with others, and the thing (object) that the concept refers to and that the sign denotes. The figure below exhibits the semiotic triangle.



A *sign* is an object that is used as a representation of a concept, e.g. a knot in a handkerchief. Symbolic signs are structures in physical substrates, which serve to carry the signs. For example, the sign "John loves Mary" can be recorded with chalk on a blackboard, and by marks on an optical disk. An *object* is an identifiable thing. We distinguish between concrete objects (e.g. a car) and abstract objects (e.g. a membership). A *concept* is a thought or mental picture of an object. Next to these, there are pure concepts, which do not refer to objects. These are *scale values* in various dimensions (length, mass, time, money etc.).

Information is the inseparable duality of a sign and the concept it designates. The sign is called the *form* and the concept is called the *content* of information. *Data* are information items. Data processing (or information processing) by ICT artifacts consists of meaning preserving transformations of signs. It is a misconception that ICT artifacts, like computers, would 'understand' information. Their 'intelligence' is really only artificial.

The ontological parallelogram

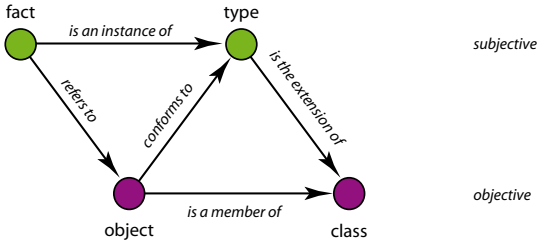
The semiotic triangle does not only apply to individual concepts, but also to generic ones. In natural languages, individual concepts are typically designated by proper names (e.g. "Ludwig Wittgenstein"), and generic concepts by nouns (e.g. "person").

A generic concept is called a type and a generic object is called a *class*. A class is the extension of a type, and a type is the intension of a class. The notion of type can best be understood as prescription of form, where form is some collection of (relevant) properties. If an object con-forms to the prescription of form of a type, then its referencing concept is said to be an instance of the type. Next, an object is said to be a member of the class that is the extension of the type to which it conforms.

In conceptual modeling, an individual concept is called a *fact*. If an object O conforms to a type T, a fact F exists, which is an instance of T. For example, there exists the fact that Ludwig

Wittgenstein is a person. Facts may also refer to multiple objects, often called aggregates. For example, the fact that Ludwig Wittgenstein is the author of the book titled "Tractatus Logico-Philosophicus" refers to a person and a book title. Each of them has its own role in he fact. Depending on the number of referred objects, we speak of unary, binary etc. facts. Accordingly, we speak of unary, binary, etc. types. Unary facts are also called *entities*.

The figure below exhibits the so-called *ontological parallelogram*, which is achieved by combining the semiotic triangle for individual concepts and the one for generic concepts. Signs are left out because they are ontologically irrelevant. This also avoids that signs and facts are confused (like mistaking the notation of a date in some calendar for the date it designates).



Sameness and change

The existence of a fact F1 starts from the moment that its object O conforms to type T1 and ends at the moment it ceases to do. If O also conforms to T2, there is also a fact F2 (which is an instance of T2). For example: (the object of) a person may conform to the type student and to the type patient. Note that a fact may come into existence when the form of an object changes and when a new type is defined.

In order to deal with changes of *composite objects*, two kinds of sameness must be distinguished: *sameness of function* (i.e. the integral whole that is capable of performing its function) and *sameness of construction* (i.e. the assemblage of its parts). For example, if some part of a car is replaced by another one, the sameness of

function still holds (it is the same car) but the sameness of construction does not (it is not the same car). Regarding composite objects, we consequently distinguish between *function type* and *construction type*. So, if a new car is produced, two entities are created: the car as a functional entity and the car as a constructional entity. The functional entity is realized by the constructional entity. Note that one may only be interested in one of them.

Conceptual modeling

The ontological parallelogram serves as the basis for building the *conceptual model* of a world. At any point in time, a world is in some state. A state is determined by the set of facts that exist at that moment. A state change or *transition* consists of one or more facts starting or ending to exist. The occurrence of a transition at some moment is called an *event* (therefore transitions are also called event types). Events are caused by acts in the system that the world is associated with. Therefore, events always concern *existentially independent* facts. Examples of independent facts are the person Ludwig Wittgenstein and the book titled "Tractatus Logico-Philosophicus". All other facts are *existentially dependent* on some independent fact. For example, the fact that the author of the book is Ludwig Wittgenstein is existentially dependent on the fact of the book title. It comes into existence at the moment the book title comes into existence.

Many entities have a *life cycle*: they start to exist at some moment and end to exist at another moment. In between their objects may conform to several *phase types* (like student and patient and adult).

The conceptual model of a world specifies its *state space* (i.e. the set of lawful states) and its *transition space* (i.e. the set of lawful sequences of transitions). The state space of a world is specified by means of the *type base* (i.e. the set of defined types) and the *existence laws*. These laws determine the inclusion or exclusion of the coexistence of facts. The transition space of a world is specified by

the *transition base* (i.e. the set of independent fact types) and the *occurrence laws*. These determine which orders of events are mandatory and which ones are prohibited. Both the state space and the transition space can be completely specified by means of (modal) logical formulas, in practice mostly represented graphically.

Illustrations of the ϕ-theory

The ontological parallelogram is the fundament of pure conceptual modeling. Among other things, it tells you that an object is a concept of some type only because the 'form' of the object conforms to the type. Therefore it is erroneous to say e.g. "let us consider this thing as a system". That is as unintelligent as saying "let us consider this professor as a parrot". The practical experiences with DEMO[2] in building world ontologies has demonstrated the value of the ontological parallelogram. Unfortunately, a lot more examples can be provided of violations of the ϕ-theory. Many of them are actually provoked by current modeling techniques and tools. For example, almost all programming languages have the type "string", next to "integer" and "real". However, string is a sign type; only integer and real are concept types. Almost all data modeling techniques, even the best ones, i.e. the fact-oriented techniques like NIAM and ORM, provoke making similar errors, by allowing sign types (lexical types) in a conceptual model.

[2] DEMO now stands for Design and Engineering Methodology for Organizations (www.demo.nl)

Violation 1
Everybody has to fill out regularly a form like the one below:

Name: *Jaap van Zweden*
Date of birth: *13-03-1957*
Address: *Mekelweg 4*
Postal code: *2628CD*
City: *Delft*
Telephone: *(015)2787822*

The pieces of text in italics are names; they designate (individual) concepts and denote (individual) objects. The type of concept or object is supposed to be mentioned before the colon. There we find some inconsistencies: "Name" and "Postal code" are no concept or object types but name types: they should be replaced by "Person" and "Postal area". Next, to be very precise, the name class should be added to every name, e.g. *"Postal code"* to *"2628CD"*. Likewise, *"13-03-1957"* is the name of a date in he name class *"Gregorian calendar"*. Dates as concepts are values on a time scale. Most programmers appear to ignore the distinction between the name of a date (in some name class) and the concept it designates. This has lead to a world wide near-disaster, known as the Y2K (year 2000) problem, generally understood as a computer error resulting from the practice of representing the year with two digits. The defense of this practice was the saving of disk storage. Let us have a closer look at it, however. The six digits of a date (ddmmyy) were stored in six ASCII characters, i.e. 48 bits. In 48 bits you can store up to $2\uparrow 48 \approx 250$ trillion dates as integers, which are all dates in about 700 billion years! Storing dates as integers would have shown that one had understood the difference between the date as a value on the time scale and the way it is named (designated) in some calendar. All Burroughs computers do this, as do all Apple Macs; they would not cause problems at the turn of the century. Preventing the Y2K disaster to occur has cost world-wide about 500 billion US dollars (almost 100 dollar per capita of the world population). So,

what these ICT professionals did (and what they still do) is not $\phi\tau\psi$ but bullshit.

Violation 2
During my stay at the University of Maastricht, I have been corrector of information modeling exams of the Open University, for several years. To my astonishment, there has not been a single one without errors. This is particularly alarming since man-weeks of professional teachers were spent on each exam. I always discovered the errors within one hour, by simply instantiating the ER (Entity Relationship) models that were provided as the solutions. The ER modeling method does not contain the generation of example instantiations for verifying the conceptual level models. Only the fact-oriented modeling methods, like NIAM (Natural language Information Analysis Method) and ORM (Object Role Modeling), include this most important step. The pioneering propagator of this verification technique is Sjir Nijssen[3].

Unfortunately, the presence of serious errors in the exams of the Open University was not an exception. Among the numerous conceptual models that I have set my eyes on, few were of acceptable quality, most suffered from various, and serious, deficiencies. It is sad to have to observe that so many are not $\phi\tau\psi$ but bullshit, as are most current (namely not fact-oriented) information modeling approaches, like ER and its derivates, including the UML (Unified Modeling Language).

Conclusion
The instantiation of (generic) concepts for checking their correctness is such an effective intellectual technique that I add it to the three techniques of Edsger Dijkstra, mentioned earlier:

4. verification by instantiation

[3] Prof.Dr.ir. G.M. Nijssen is the spiritual father of NIAM.

By understanding and applying this technique, next to the other three, the design of information systems becomes intellectually manageable. The elementary fact, expressed in an elementary sentence, is truly the basis of conceptual modeling, as Wittgenstein already demonstrated in his Tractatus Logico-Philosophicus.

Larghetto assai

In the 90's of the 20th century one came to realize that the ultimate goal of applying ICT should be to improve the operation of enterprises and to let ICT artifacts be an integral part of their organization. A growing need for agility of enterprises supported this idea. To achieve the goal, however, one needs a thorough understanding of systems and of the way they can be integrated.

Below, the τ-theory is summarized (the Greek letter "τ" is pronounced as "TAO", which are the first letters of "Technology", "Architecture", and "Ontology"). It is a theory about the development (i.e. design, engineering, and implementation) of artifacts of all kinds. It helps to pursue the said goal by providing deep answers to key questions like: What is the difference between function and construction? How does this distinction affect the design process? What is Technology, Architecture, and Ontology? What role do they play in system development? The τ-theory is rooted in systemics, in ontology (including mereology), and in design theory.

System

The *teleological system* notion is concerned with the function and the (external) behavior of a system. It is adequate for the purpose of *using* or *controlling* a system. The *ontological system* notion is concerned with the construction and the (internal) operation of a system. It is adequate for the purpose of *building* or *changing* a system. The ontological system notion is defined as follows: something is a system if and only if it has the following properties:

Composition: a set of elements that are atomic in some category (physical, social, etc.).

Environment: a set of elements of the same category; the composition and the environment are disjoint.

Production: the elements in the composition produce things (for the benefit of the elements in the environment).

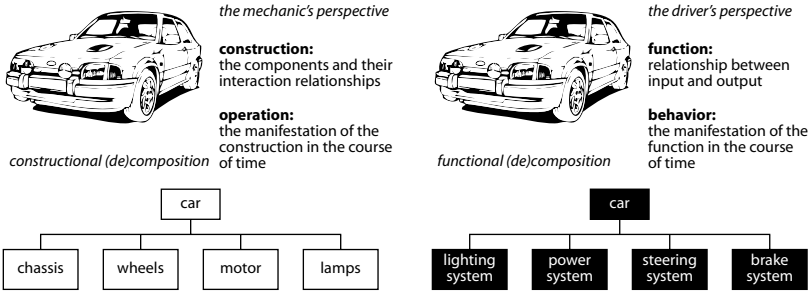
Structure: a set of influencing bonds among the elements in the

composition, and between them and the elements in the environment. We distinguish between active influencing, called *interaction*, and passive influencing, called *interstriction*.

The elements in the composition and the environment are atomic with respect to the system category. For example, the elements of a social system are social individuals, or subjects (human beings). The effect of the production of a system is conceived as state changes of the system's world, as is the effect of interaction. Interstriction consists of inspecting and taking into account the state of the system's world by the elements in the composition when being active.

Model

A *white-box* (WB) model is a direct conceptualization of the ontological system definition. It represents the *construction perspective* on systems. A *black-box* (BB) model represents the *function perspective* on systems; it is actually identical to the teleological system notion. Below, the white-box model (left) and the black-box model (right) of a car are exhibited.



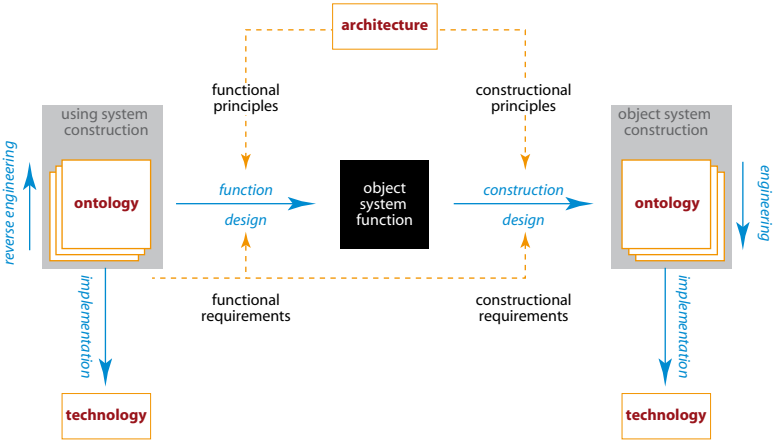
Constructional (de)composition and *functional (de)composition* are fundamentally different. The first one applies to the car as a constructional entity and the second one applies to the car as a

functional entity. There is only one constructional (de)composition, but there may be as many functional (de)compositions as there are subjects that conceive them. This follows from the distinction between functional entity and constructional entity (cf. the ϕ-theory).

The *purpose* of a system is not an inherent system property but a relationship between the system and a stakeholder. The *function* of a system is a socially agreed upon purpose.

The Generic System Development Process

In engineering, the *use-support* structure between systems is an important one. To say that a system S1 uses a system S2 (and conversely S2 supports S1) means that the construction of S1 uses the function of S2, which by nature is expressed in the 'language' of the construction of S1. A use-support relation between two systems is neither a sub-super nor a level structure relation, but a sui generis relation.



The figure above exhibits the complete and generic process for developing an *object system* (OS) for the sake of supporting a *using system* (US). It consists of four phases: function design, construction design, engineering, and implementation. *Function design*

starts from the construction of the US and results into the functional specifications (functional model) of the OS. *Construction design* starts from here and results into the *ontological model* of the OS. This model is *engineered* into the *constructional specifications* (implementation model) of the OS. Function design and construction design are generally iterative, in order to arrive at a balanced compromise between reasonable functional specifications and feasible constructional specifications.

Ideally, function design starts from the *ontology* (or ontological model) of the US. This is defined as the highest level constructional model, which is fully implementation independent. This is the only way to arrive at objectively determinable (functional and constructional) requirements. If the ontological model of the US is missing, it can be produced through *reverse engineering*. Next, it is of paramount importance that the functional specifications of the OS do not contain any constructional issue.

Ideally, construction design starts with the ontological design of the OS, followed by engineering (also called *technical design*). This means that one first analyzes the functional 'problem' completely in implementation independent constructional terms (the ontology of the OS) before going to synthesize the 'solution'. Only in this way can one objectively guarantee that an optimal *implementation model* (the lowest level constructional model) will be delivered.

Theoretically, *architecture* is the normative restriction of design freedom. Practically it is a coherent set of *design principles*, derived from and consistent with the strategy of the US. These principles can be divided into *functional* and *constructional* principles. They come in addition to the requirements and therefore may be conceived as the operational specification of generic requirements (next to the specific requirements regarding a particular OS).

Illustrations of the τ-theory

Apparently, it is very hard for the human mind to make and maintain a sharp distinction between function and construction. For sure, one of the reasons is that in the natural languages purely constructional names are very rare. Making the distinction carefully in the process of designing is however of utmost importance. Christopher Alexander[4] provides the excellent example of the design of the one-hole kettle, which was 'invented' in the 50's of the 20th century. Imagine that you would have the task to design a new kettle according to, among others, the next functional requirements: 1) there must me a hole to pour water in, and 2) there must be a hole to pour water out. Consider how difficult it is to come up with only one constructional element that satisfies both requirements if you are only familiar with two-hole kettles. Note also that, as a consequence, design patterns are potentially counterproductive since they deny the need for creativity in constructional design.

Like in enterprise engineering (as we will see later), there is an urgent need in software engineering for a theory that is founded on truly elementary notions of data processing, and such that combinatorial explosions of software changes can be avoided. Normalized Systems provides such a theory[5].

Violation 1

Twelve years ago, the newspaper NRC Handelsblad (Nederlands), showed a picture of Garry Kasparov in desolation during his last game against Deep Blue, on 11 May 1997. As you know, Kasparov lost the game and by that the whole match. The caption of the picture was "Wereldkampioen verkwanselt waardigheid" (in WoLa: "World champion fritters away dignity"), which is very unfair. What happened on that day had to happen some day. Let me draw an analogy for the sake of argument. I have lived in the same street

4 Alexander, C.: *Notes on the Synthesis of Form*, Ablex Publishing Company, New Jersey (USA), 1960

5 Normalized Systems is developed by Prof.Dr. H. Mannaert and Prof.Dr. J. Verelst.

as Piet van der Kruk. Piet van der Kruk is multiple Dutch champion in weightlifting; he represented the nation at the Olympics of 1968. Suppose the picture showed Piet van der Kruk, with the caption "Piet van der Kruk verslagen door vorkheftruck" (in WoLa: "Piet van der Kruk beaten by forklift truck"). That sounds hilarious, indeed. Of course, forklift trucks can lift larger weights than humans can.

In 1997 many chess players thought this is the end of the sport, or that they would be beaten by chess computers in tournaments from now on. They seem to have forgotten that forklift trucks have never been admitted to the Olympic games. The confusion comes from not understanding the distinction between function and construction. As we have seen, function is in the eye of the beholder; it is not a system property. Therefore, Deep Blue did not play chess, it only computed, along predetermined paths. Likewise, forklift trucks do not lift weights, they only move in predetermined ways. Anthropomorphizing inanimate objects is as dangerous as it is human. In science, it is a symptom of professional immaturity, as Edsger Dijkstra has pointed out[6]. So, the fuss about Deep Blue was not $\phi\tau\psi$ but bullshit.

My suggestion to chess players is to use computers for improving their performance, like race-drivers use cars to run faster. I am pretty sure that if both are equipped with the same (very) Deep Blue, Garry Kasparov can easily be beaten by Jaap van den Herik[7].

Violation 2

In the 70's and 80's of the last century a lot of methodologies were developed to help manage the complex task of applying ICT to organizations, i.e. to design, engineer, and implement ICT applications. In the 90's they were renamed into architecture frameworks, often without appreciable modifications or extensions. In order to

6 Prof.Dr. E.W. Dijkstra made this statement in EWD1284.
7 Prof.Dr. H.J. van den Herik is well-known for his research in computer chess.

avoid that the providers of the numerous commercial architecture frameworks become offended (or that I forget one), I choose TOGAF, the framework of the Open Group, for closer investigation. TOGAF violates the τ-theory in many ways. First, it lacks any scientific foundation. Second, it does not distinguish clearly between function and construction, nor between functional and constructional design. Third, it does not apply the notion of system ontology, nor a sensible notion of architecture; instead it uses the term "architecture" to mean several different things. In summary, TOGAF is an incoherent and inconsistent collection of best practices.

In spite of all these troubles, there are numerous certified TOGAF architects in the world. I am unable to imagine what sensible meaning that qualification could have. So, TOGAF is not $\phi\tau\psi$ but bullshit. Recall that the same holds for the other architecture frameworks.

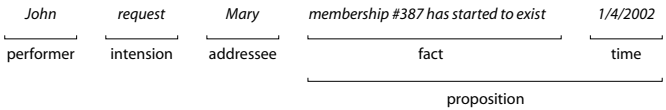
Appassionato con brio

Next to the thorough understanding of systems, as contained in the τ-theory, a second paradigm shift was needed in the 90's of the 20th century, in order to bridge the gap between information systems engineering and the organizational sciences. The new paradigm is brought to us by the social sciences and by language philosophy. It is the insight that communication is not exchanging information but primarily social interaction: while communicating, people engage into commitments.

Below, the ψ-theory is summarized (the Greek letter ψ is pronounced as PSI, which stands for Performance in Social Interaction). It bridges the said gap and provides deep answers to key questions like: What is the common core of information, action and organization? How can the immense complexity of organizations be made intellectually manageable? The ψ-theory explains how and why people cooperate, and in doing so bring about the business of an enterprise. It is rooted in the information sciences, notably semiotics and language philosophy, in systemics, and in the social sciences, notably communicative action theory.

Organizational Atoms

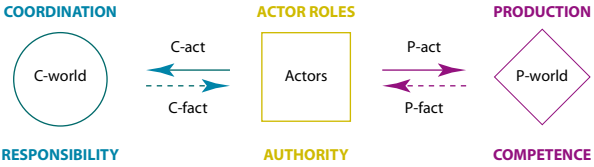
An *organization* is a social system, i.e. a system of which the elements are subjects (human beings in their capacity of social individual). They perform two kinds of acts. By performing *production acts* (or P-acts for short), the subjects contribute to bringing about products and services. A P-act can be material (like manufacturing) or immaterial (like judging). By performing *coordination acts* (or C-acts for short), subjects enter into and comply with commitments towards each other regarding the performance of P-acts. The generic structure of C-acts is exhibited in the figure below. The example concerns the request by the subject John (the performer) towards the subject Mary (the addressee) to become member, e.g. of a library.



The *proposition* consists of a P-fact (in this case the start of a membership) and an associated time (in this case the start date of the membership). The *intention* represents the 'social attitude' taken by the performer with respect to the proposition (in this case the request). In order to perform the exhibited C-act successfully, three conditions must be satisfied, for which John and Mary have to perform a number of communication acts. The most important condition is the *performa condition*, which means that they have to achieve social understanding. The condition is satisfied if in Mary's mind the *commitment* is evoked to respond appropriately to John's request. A prerequisite for this is satisfying the *informa condition*, which means that they have to achieve intellectual understanding. This is the case if the right conceptions of both the intention and the proposition are induced in Mary's mind. Satisfying the informa condition needs at least two communication acts: one in which John informs Mary, and the other one in which Mary confirms to have understood him. A prerequisite for this is satisfying the *forma condition*, which means that they have to achieve significational understanding. This is the case if for every communication act at the informa level the sentence uttered by John is transmitted without distortion to Mary and correctly perceived by her, and vice versa.

The notion of *actor role* serves to abstract from the subject that performs an act. It is defined as the *authority* to perform a particular kind of P-act (and the corresponding C-acts). An actor is a subject in its fulfillment of an actor role. Authority is related to *competence* and *responsibility*, as exhibited in the figure below. As the result of performing a C-act, a C-fact is created in the C-world. Likewise, by performing a P-act, a P-fact is created in the P-world. While performing C- and P-facts, actors take into account the

current state of the C- and the P-world (represented by the dashed arrows).



The occurrence of a C-fact is a *business event* to which some actor is committed to respond. Since they include the P-act/fact it is about, C-facts are the atomic building blocks of organizations, next to the *business rules* that actors apply in dealing with business events.

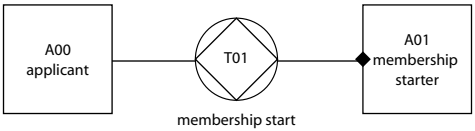
The notion of intention (in communication) is the key to the coherent understanding of information, action and organization: the ψ-theory provides the needed new paradigm.

Organizational Molecules

C-acts and P-acts occur in a universal pattern, called *transaction*. A transaction involves two actor roles; one is the *initiator* and the other one the *executor*. The initiator starts a transaction by requesting the P-fact and ends it by accepting the delivered result. The executor promises to perform the P-act and states the achieved result. This constitutes the basic transaction pattern. The complete, universal, pattern consists of twenty different C-acts. Every transaction process is a path (possibly including iterations) through the *universal transaction pattern*. Ideally, C-acts are performed *explicitly*. They may however be performed *implicitly*, and even *tacitly* (which means that there is no observable act at all). The executor of a transaction is considered to be the *owner* of the produced P-facts.

An *elementary actor role* is the authority to be the executor of exactly one transaction kind (of which the result is one kind of P-facts, e.g. starting library memberships). Transaction kinds with

their corresponding elementary actor roles constitute the molecular building blocks of organizations (see the figure below).



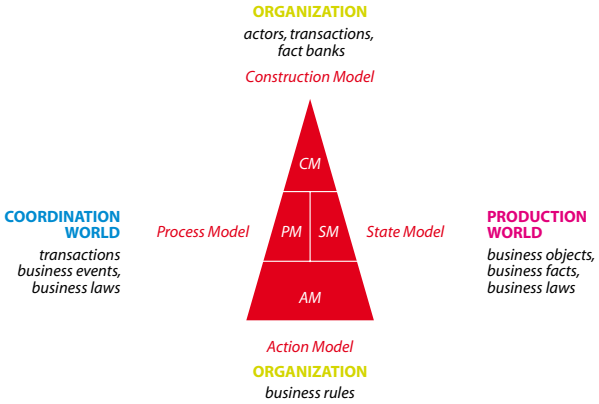
The executor (A01) of transaction kind T01 may on its turn be the initiator of one or more transaction kinds. The same reasoning holds for the executors of these transaction kinds. In this way a tree structure of transactions is constructed, called *business process*. By understanding business processes as tree structures of organizational molecules, a considerable reduction of complexity is achieved (in practice well over 70%).

Enterprise Ontology

Every organization is a layered integration of three aspect organizations, called its B-organization, its I-organization, and its D-organization, in accordance with the use-support structure (see the τ-theory). The *B-organization* comprises all transaction kinds (and their corresponding actor roles) that deliver business services, i.e. of which the P-act/fact is a business act/fact. These include the transactions that deliver such a service to the environment as well as the transactions that deliver components for such a service to internal actors. The *I-organization* comprises all transaction kinds (and their corresponding actor roles) that deliver informational services to the B-actors. An *informational service* concerns the remembering and the provision of B-organization P- or C-facts, including externally acquired P- or C-facts, or of derived facts based on these. The *D-organization* comprises all transaction kinds (and their corresponding actor roles) that deliver documental services to the I-actors. A *documental service* concerns the creation, the transportation, the storage, or the retrieval of a document that contains information needed by the actors in the B-organization. So, the

D-organization supports the I-organization, as the I-organization supports the B-organization.

The understanding of the B-organization of an enterprise according to the ψ-theory is called the *ontology* of the enterprise. By focusing on the B-organization of an enterprise, another considerable reduction of complexity is achieved (in practice well over 70%). As is exhibited in the figure below, there are four distinct views on the ontology, or ontological model, of an enterprise, called the construction model (CM), the process model (PM), the state model (SM), and the action model (AM).



Illustrations of the ψ-theory

The ψ-theory is probably the most innovative of the three theories, as well as the most revolutionary. Although the need for a coherent, consistent, comprehensive and yet concise understanding of the essence of an enterprise is recognized widely, few professionals use a methodology with such characteristics. If they would do, we would not have anymore incompatible process models and data models. We also would not have anymore enterprise 'architectures' in which the business 'architecture' part only contains a discussion of business goals and strategies. Rarely does a business 'architecture' contain conceptual models of the enterprise that could serve as an appropriate starting point for producing the information (systems)

'architecture', as it should be the case. Lastly, we also would not call banks and insurance companies information processing enterprises, for the only reason that they have no tangible products. Immaterial B-organization P-facts are as real as material ones!

At the same time, one of the key success factors of all DEMO-projects has been the presentation and discussion of the enterprise ontology, thus the ontological model of the B-organization. Particularly managers like it, because they are concise and they 'speak their language'. Often the construction model is sufficient for analyzing the problems at hand, and for finding satisfying solutions.

Violation 1

When the grandfather whom I had known very well died, I could only control my tears by telling myself that there were just bones and flesh and skin inside the coffin. Later I learned that what I did is called reductionism in philosophy. Reductionism can be an appropriate intellectual technique but it is also rather dangerous since it hides the essence. Sure, there were bones and flesh and skin inside the coffin but that was not the cause of my grief. I had lost my living grandfather, for ever! That was my real sorrow, but such a sorrow is too much for a child to bear.

Reductionism is applied on a wide scale in business process modeling. All well-known current approaches to business process modeling, like Flowchart, BPMN (Business Process Modeling Notation), ArchiMate, EPC (Event Process Chain), and PetriNet, reduce business processes to sequences of (observable) actions and results, and human beings to human resources. No distinction is made between the B-, the I-, and the D-organization. Moreover, the resulting models lack the transaction structures, which makes them completely useless for redesign and re-engineering, if only for the fact that they are unable to detect tacitly performed coordination acts (which happen to occur quite often). So, contemporary business process modeling is not $\phi\tau\psi$ but bullshit. One better erase the predicate "business".

Let me add an anecdote for clarification. About a year ago, two representatives of the Nederlands Police visited me, actually to complain about the process models that had been produced by a Nederlands company in the past three years, covering all business processes of the national police. They wanted to use these models for re-designing and re-engineering the processes and for deriving information requirements for supporting ICT applications, but they had repeatedly failed to do so. I had warned them before (which they admitted). Now, I could only advise to put the whole stuff in the trash and start again, but properly. To worsen the case, these models were produced in a variant of SADT/IDEF0[8], which is a function-oriented modeling technique. Consequently, they reflect the personal interpretation of the modeler. Do I go too far in considering this malperformance a societal crime?

Violation 2

Some time ago I overheard, while waiting at the reception of a furniture factory, a discussion on the telephone between the receptionist and a client. Apparently, the client wanted to change the color of the sofa that he had ordered the day before. The receptionist could only fully agree that the client's whish was quite normal and should be fulfilled. However, she was unable to help him because the ICT application did not allow for such a change. She had to tell the client that he would have to wait until the sofa was delivered at his house and then return it (sic!). Is the design of this business process (or of the ICT application) $\phi\tau\psi$? No, this is bullshit. Unfortunately, the case is not an exception. Business process and/or application designers seem to be unaware of the natural human way to conduct business, thus of the universal transaction pattern. What the client needed was the option to cancel his request, something that is most human.

[8] SADT stands for Structured Analysis and Design Technique. IDEF stands for Integrated DEfinition for Function modeling.

Another illustration of the same point regards the design of the user interface of elevators. Apart from the silly practice that one has to 'split up' one's request to be transported to some floor in two phases (first: up or down, then: the specific floor), a really nasty flaw is that almost all elevators do not offer the possibility to correct a mistake in the choice of the floor. At the time I was studying at the TU Eindhoven, the elevators in the main building allowed for it. They had push-buttons for requesting to be transported to a particular floor; these buttons could also be pulled out again, by which the request was cancelled! Presently, the TU Eindhoven has modernized the elevator system; it is not possible any more to cancel a request. Is this progress?

Conclusion

The ontological model of an enterprise offers a total reduction of complexity of well over 90%. Moreover, it is coherent, consistent, and comprehensive. Next, it reflects the way social individuals (human beings) behave when cooperating in order to achieve some (business) goal. Therefore, validating the design of a business process or an ICT application on the basis of the ontological model is a prerequisite for delivering anything useful at all. This intellectual technique has proven to be so effective that I add it to the four techniques we have collected up to now:

5. validation from ontology

By understanding and applying this technique, next to the other four, the design of organizations becomes intellectually manageable.

Grande finale

I was born on 20 June 1945 in Brunssum (Nederland). From the age of 1 and 10 and 11 and 100 I can't remember anything, psychologists tell me. They seem to be right: I can't remember my birth at all. The two facts about it that I mentioned, I know only from hearsay. However, I am pretty sure that I can remember me rocking for hours on the wooden rocking horse in my house at the age of 100, preferably early in the morning when the others were still sleeping. When I was 101 I built my first, in my view very complicated, machines with Meccano.

At the age of 1000 I was a shy boy who believed everything he read. I spent my pocket money on buying bitter macaroons (which I didn't like at all) after I had read the Nederlands saying "bitter in de mond maakt het hart gezond" (in WoLa: "bitter in the mouth makes your heart healthy"). I also started fretsawing table lamps and constructing low-voltage electrical circuits. When I was 1001 I stopped believing everything and consequently to buy bitter macaroons, and I became an altar boy. Two years later I got fired, after having dropped the missal during a high mass.

At the age of 10000 I was still a shy boy. I got my glider license and I drove former military jeeps to transport sailplanes on the airfield and to pull out cables. I also operated the winch to pull the airplanes in the sky. All this thanks to my father who was gliding instructor and who took me with him in the sky since I was ten. Next to that, I spent a lot of time on building and maintaining a large model railway together with my elder brother. When I was 10001 I fell terribly in love with a girl; she was soon taken over by my elder brother. Even my mother was not able to comfort me. I also got my diploma Gymnasium-B from the St. Thomascollege in Venlo.

At the age of 100000 I thought I had made it to the top. I was married with a loving woman, had two cute daughters, and lived in a large house with a huge garden in the countryside. When I was 100001 I fell into a deep crisis. For half a year I was not able to work. I was being treated by a psychiatrist, used antidepressants, and participated in a number of counseling groups, both mixed and men only. After a year I climbed out of the dumps as a more sensitive, social, and self-confident person; and definitely atheistic, which made me more open-minded, peaceful, and independent; and firmly determined to leave industry and to look for a position in academia. I wanted to understand deeply the problems I had experienced in applying ICT in enterprises, and I wanted to contribute to their solutions.

Now, at the age of 1000000 I am still a shy boy ... sometimes. I look back with satisfaction at a life that counts few dull moments. Next year, when I become 1000001, I will get my AOW (a social pension in Nederland for everyone above 65).

It is very unlikely that I will speak to you again at the age of 10000000, because most of you will not show up, I'm afraid. However, I hope to enjoy every day of the rest of my life amid my business friends in the DEMO Knowledge Center[9], which currently counts well over 200 certified DEMO Professionals, amid my scientific friends in the CIAO! Network[10], which currently counts five member institutes[11] and five candidate institutes[12], and together with my family, in particular my five children and their partners, my five grandchildren, and of course my partner in life, Gerdien, with her two daughters.

I want to thank the TU Delft, the faculty of EEMCS, and the department of Software Technology for having allowed me to serve as full professor for the past 15 years. Special thanks go to the col-

[9] www.demo.nl or www.ee-institute.org
[10] www.ciaonetwork.org
[11] TU Delft, Utrecht University of Applied Sciences, University of Antwerpen, TU Lisboa, University of St. Gallen.
[12] Cnam Paris, State University of Moskwa & Nishnii Novgorod, Tokyo Institute of Technology, University of Hohenheim (Stuttgart)

leagues in the department, and the students, for supporting me unconditionally after I had been told, on 1 February 2002, that my chair would be discontinued, in the shortest meeting I ever have had with the then dean. I had no answer to the wrongness and the unfairness of the decision, and to the clumsiness with which it was made known to me. The more so, since a few hours before my youngest sister had called me to tell that she had breast cancer. Wiped out by the stroke of a key. Fortunately, my sister is sitting in the audience, in excellent health. The rest of that Friday afternoon I spent sitting at my desk, overlooking my life, crying, and becoming convinced that this would not make me go away. I had not exchanged Maastricht for Delft in order to leave through the backdoor. I had a mission to fulfill, the mission of enterprise engineering.

I have failed in establishing enterprise engineering firmly at the TU Delft, but I have been successful in disseminating DEMO in practice, thanks to the many supporters I have got in the course of time. This support is remarkable for an environment that needs some amount of conservatism to survive, because DEMO (and for that matter the $\phi\tau\psi$-theory) is a Copernican revolution, as several practitioners have told me after their DEMO training. In contrast, the scientific world has been much more keeping off, which is remarkable the other way around. I consider the current peer review system for scientific quality control a major obstacle to the dissemination of sound innovative ideas, since it is dominated by the 'do as we do' syndrome.

My mother often told me that I must have a special guardian angel. I do, I have even more than one. The most long-lasting guardian angel is my friend Clemens Lasance. He always tries to keep me on the right trail, with success. Three new guardian angels appeared on the stage during my stay in Delft, in this order: Hans Mulder, Jan Hoogervorst, and Antonia Albani. They keep motivating me and helping me in pursuing our common mission, and not to loose heart when the umpteenth nitwit refuses even to attempt

to understand $\phi\tau\psi$ because of it being too different from what he was used to. The four of us constitute the 'think tank' called CIAO! Symposium[13].

Although I have failed in my mission to establish Enterprise engineering at the TU Delft, I am fully confident that one time it will be, as a result of the activities of the CIAO! Network. I envision a future in which enterprise engineers are educated in Utrecht, Antwerpen, Lisboa, St. Gallen, Paris, Moskwa & Nishnii Novgorod, Tokyo, and Stuttgart, next to Delft and to many other places. I also envision that these enterprise engineers will succeed in convincing other academic professions of their indispensable high societal value, as well as of the high societal value of engineers in general.

A last advise to the students (whom I will miss): stay always faithful to $\phi\tau\psi$ and recall daily these truthful statements:

"There is no instant wisdom, there is only painstaking scientific and philosophical research"
(Mario Bunge)

"If you think you understand something, you have not thought about it enough"
(Richard Feynmann)

Next, keep improving you professional abilities, or become one of those rare but badly needed real managers, who understand that people are the gold of any enterprise and thus do not degrade human beings to human resources; who do not strive to meet targets but to let people develop their talents, while having their organization designed, engineered and implemented by enterprise engineers such that it is most effective, efficient, and agile, knowing that targets then will be met as a side effect. In summary: don't bullshit and don't let being bullshitted because:

[13] The original meaning of the Greek word (which is: drinking together) should be taken seriously, but not too seriously.

"Bullshit is a greater enemy of truth than lies are"
(Harry Frankfurt)

Planes must fly, not crash!

Ik heb gezegd.

APPENDIX -

selected books in Enterprise Engineering (chronologically)

Reijswoud, V.E., *The Structure of Business Communication – Theory, Model and Application*, dissertation TU Delft, 1996

Steuten, A.A.G., *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*, dissertation TU Delft, 1998

Hommes, L.J., *The Evaluation of Business Process Modeling Techniques*, dissertation TU Delft, 2004

Renssen, A.S.H.P. van, Gellish – *A Generic Extensible Ontological Language*, dissertation TU Delft, 2005

Shishkov, B.B., *Software Specification based on Re-usable Business Components*, dissertation TU Delft, 2005

Dietz, J.L.G., *Enterprise Ontology – Theory and Methodology*, Springer Verlag, 2006

Mulder, J.B.F., *Rapid Enterprise Design*, dissertation TU Delft, 2006

Op 't Land, M., *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*, dissertation TU Delft, 2008

Dietz, J.L.G., *Architecture – building Strategy into Design*, Academic Service (Sdu), 2008

Hoogervorst, J.A.P., *Enterprise Governance and Enterprise Engineering*, Springer Verlag, 2009

Mannaert, H. Verelst, J., *Normalized Systems – Recreating Information Technology Based on Laws for Software Evolvability*, Koppa, Belgium, 2009